Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Semiannual Technical Report
covering the period
September 1, 1972 - January 15, 1973

Form Approved Budget Bureau No. 22-R0293

Sponsored by
Advanced Research Project Agency
ARPA Order No. 1997

ARPA ORDER Number:
1997/12-02-71

Program Code Number:
80230

Contractor:
Massachusetts Institute of Technology
Cambridge, Mass. 02139

Contract Number:
N00014-67-A-0204-0064

Principal Investigator:
Alan V. Oppenheim
617-253-4177

Scientific Officer:
Director, Information Systems
Mathematical and Information Sciences Division
Office of Naval Research
Department of the Navy
800 North Quincy Street
Arlington, Virginia 22217

Effective Date of Contract:
January 15, 1972

Contract Expiration Date:
January 14, 1974

Amount of Contract:
$96,377.00

Short Title of Work:
Speech and Picture Processing

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

2a. REPORT SECURITY CLASSIFICATION: Unclassified

2b. GROUP: None

3. REPORT TITLE

Semiannual Technical Report

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Progress Report for period ending January 15, 1973

5. AUTHOR(S) (First name, middle initial, last name)

J. Allen

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February 28, 1973	8	4

8a. CONTRACT OR GRANT NO.

ONR Contract N00014-67-A-0204-0064

b. PROJECT NO.

ARPA Order No: 1997/12-71

c.

Program Code No: 80230

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

None

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Advanced Research Projects Agency
of the Department of Defense

13. ABSTRACT

During the second half of the contract year the program continued the following studies: speech analysis by linear prediction, reconstruction of multidimensional signals from projections, development of a high speed digital processor for speech synthesis, and the design of two-dimensional recursive digital filters. These projects are summarized, and reprints of available publications are appended.

A

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

304050

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Speech analysis						
Linear prediction						
Digital filters						
Multidimensional signals						
Recursive digital filters						
High-speed digital processor						

Semi-Annual Technical Report
ONR Contract N00014-67-A-0204-0064
covering the period
September 1, 1972 - January 15, 1973
submitted by
J. Allen
(Acting Principal Investigator)
February 28, 1973

Projects Studied Under the Contract

During the second half of the contract year (Jan. 15, 1972 - Jan. 15, 1973), the program continued the following studies: speech analysis by linear prediction, reconstruction of multi-dimensional signals from projections, development of a high speed digital processor for speech synthesis, and the design of two-dimensional recursive digital filters. These projects are summarized in the following pages. Reprints of available publications are appended.

The views and conclusions contained in this document are those of the author, and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

1. Speech Analysis by Linear Prediction

Up to the present, most of the effort has been devoted to the development of an interactive speech analysis system, which is implemented on the Fast Digital Processor. The analysis procedure is based on the technique of linear predictive analysis. In this scheme, the combined effect of glottal source, vocal tract, and radiation losses is represented by a single all-pole filter. In this way, spectral frames are constrained to have a fixed number of resonant peaks, which are located at the positions of the formants in conventional speech analysis. These spectra are smooth, yet they retain the important information about the formants. The implemented system allows for the rapid and interactive analysis of speech

samples, and the convenient storage of the computed spectral frames. It is possible to mark and edit the data in flexible ways, and to compute and attach auxiliary calculations to the basic data. Any of these computed parameters can be easily displayed and examined.

The system will be used to create a very large data base of processed utterances for use in speech recognition research, which is the next phase of this project. Bisyllabic utterances of the form /h$\partial C_1 V C_2$/ will be recorded for all possible C_1, C_2 combinations, and for a wide range of the vowels V. The goal will then be to describe the acoustic phonetic parameters of C_1 in these utterances. While there are many known contextual effects on the phonetic realization of sounds, it is felt that these are minimized for consonants in pre-stressed position. Furthermore, stressed syllable nuclei are reliable anchor points at which to initiate phonemic recognition. For these reasons, recognition of pre-stressed consonants can be expected to be at least as reliable as that of consonants in other positions, and hence a substantial effort toward their recognition is justified.

We expect that this study will represent a comprehensive attack on this problem, leading to the creation of an exceptionally large data base, together with a wide range of techniques for consonant recognition and a critical evaluation of their capabilities.

2. Reconstruction of Multidimensional Signals from Projections

In many applications, a set of projections of an N-dimensional object onto (N-1) dimensions are available from which it is possible to reconstruct the original object. X-ray photographs, as obtained in medical applications and the inspection of mechanical systems, represent two-dimensional projections of the three-dimensional objects which have been X-rayed. A number of new results for this problem have been obtained by formulating the reconstruction problem directly in digital signal processing terms. Based on this formulation, several algorithms have been developed which appear, based on several reconstruction examples, to be superior to previous algorithms

in some cases. Most of the reconstructions performed have been the transformation of one-dimensional projections to two-dimensional photographs. A reconstruction of a section of leg bone from real x-ray data has also been performed. This work has resulted in the completion of an Sc.D. thesis (R.M. Merserau; "Digital Reconstruction of Multidimensional Signals from their Projections"), the abstract of which is attached. Some of the algorithms are also summarized in the paper "The Digital Reconstruction of Multidimensional Signals from their Projections" by R.M. Merserau; Proc. 10th Annual Allerton Conference on Circuit and System Theory, Oct. 4-6, 1972, Monticello, Ill., pp. 326-334.

Some preliminary investigations into the use of projections for picture bandwidth compression have also been completed, which have led to promising results. If a function can be represented by its projections, then perhaps pictures can be transmitted or stored by utilizing a set of projections for these pictures, thus using fewer bits than are required for transmitting the entire picture directly. These experiments are summarized in the doctoral thesis by Merserau.

3. Development of a Digital Processor for Speech Synthesis

Detailed design of the "Black Box" processor has been completed. There have been two major changes in design, and several smaller ones as the specific logic has been developed. First, a major decision has been made to build the processor from ECL 10k logic, rather than 74 series and Schottky TTL. This change will increase the speed of the processor, but it should also result in greater system reliability. It is our belief that with proper design precautions as currently understood, ECL systems should be more reliable and immune from noise problems than TTL systems. The second major change has been an increase in data word length from 18 to 24 bits to permit retention of more significant figures. The new multiplier will be 16 X 24, and should run in less than 100 nsec. The memory uses 1024X1 ECL chips which are just now becoming available, and which

provide adequate storage at state-of-the-art speeds.

Large circuit boards have been ordered, and ECL parts will be ordered shortly. The only remaining design problems concern the exact nature of the PDP-9 interface, which must now be modified to allow for 24-bit data words.

An internal document, "The Black Box", is attached which describes the design of the processor in detail.

4. Design of Two-Dimensional Recursive Digital Filters

Recent work has been based on using the one-projection results of Merserau to allow the inference of two-dimensional structures from their one-dimensional projections. In this way, one-dimensional approximation theorems can be used to simplify the two-dimensional recursive approximation problem. The theory for the design of two-dimensional recursive filters whose magnitude-squared frequency response approximates a desired function is complete, and the algorithm has been programmed and is currently being debugged. There are, however, theoretical problems concerning the realizability of the designed filters. These problems, which must be circumvented before the design algorithm can be considered complete, are the focus of current effort.

Additionally, a simple inverse filtering program was written using the FDP-Univac picture processing system developed last year. Digital photographs have been blurred by a recursive lowpass filter (taking approximately 10 seconds), and then reconstructed exactly using a non-recursive high-pass filter (taking approximately 5 seconds).

Publications

1. R.M. Merserau "Digital Reconstruction of Multidimensional Signals from their Projections" Sc.D. thesis, Dept. of Electrical Engineering, M.I.T., January 17, 1973.

2. R.M. Merserau "The Digital Reconstruction of Multi-dimensional Signals from their Projections" Proc. 10th Annual Allerton Conference on Circuit and System Theory, Oct. 4-6, 1972, Monticello, Ill., pp. 326-334.

3. J. Allen, F.X. Carroll, E. Jensen "The Black Box" internal memorandum.

4. A. V. Oppenheim and C. J. Weinstein, Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform (Proc. IEEE Vol. 60, No. 8, pp. 957-976, August 1972)

DIGITAL RECONSTRUCTION OF MULTIDIMENSIONAL SIGNALS FROM THEIR PROJECTIONS

by

Russell Manning Mersereau

Submitted to the Department of Electrical Engineering on January 17, 1973 in partial fulfillment of the requirements for the Degree of Doctor of Science.

ABSTRACT

Several algorithms for the reconstruction of multidimensional signals from their projections are presented. These algorithms can be applied to the problem of estimating the structure of an unknown three-dimensional object from its x-ray photographs or electron micrographs taken at different orientations. The reconstruction problem is broken into two distinct steps; first samples of the Fourier transform of the unknown signal are computed from a series of digitized projections, then the unknown is estimated from the samples of its Fourier transform. Reconstructions are considered from several sets of samples in Fourier space. A particular set of samples, the concentric squares raster is developed, the reconstructions from which are superior to those made from the more traditional polar raster of samples for bandlimited inputs which have a rectangular frequency band. Furthermore for an important class of unknowns exact reconstructions can be performed from a concentric squares raster from a finite number of projections. In fact for this class of unknowns a single projection is sufficient. A detailed treatment of the one-projection reconstruction problem is presented and the difficulties associated with its solution are explored.

THESIS SUPERVISOR: Alan V. Oppenheim
TITLE: Associate Professor of Electrical Engineering

The Digital Reconstruction of Multidimensional Signals from Their Projections

by

R. M. Mersereau

Presented at

Tenth Annual Allerton Conference on Circuit and System Theory

October 4-6, 1972

Allerton House, Monticello, Illinois

Sponsored by the

Department of Electrical Engineering and the
Coordinate Science Laboratory of the
University of Illinois at Urbana-Champaign

THE DIGITAL RECONSTRUCTION OF MULTIDIMENSIONAL SIGNALS FROM THEIR PROJECTIONS.

RUSSELL M. MERSEREAU

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

Several algorithms for approximating a multidimensional density function in terms of its projections ("x-ray photographs") at different orientations are presented. This is accomplished by means of a theorem which states that the Fourier transform of a projection of a function is a slice (central section) of the Fourier transform of that function. Special emphasis is given to bandlimited functions as these are readily digitized. Some theorems are presented which state that bandlimited functions of a certain class can be represented by a single projection.

INTRODUCTION

There are occasions when the structure of a three-dimensional object is unknown and desired but only two-dimensional projections of that object are available. A transmission x-ray photograph might represent such a projection. A single x-ray photograph, since it is merely a two-dimensional representation of an inherently three-dimensional structure, does not contain all of the information that a physician might want, since all detail in a direction normal to the photographic plate has been superimposed. One solution to this dilemma is to take x-ray photographs from different vantage points and use this set of x-rays to recreate a three-dimensional image, say in a digital computer. In this paper we will consider several techniques for accomplishing this. In addition to performing reconstruction from x-rays [4] [5][7][8], these algorithms are useful for reconstructing from electron micrographs [2][3], fan-beam radio telescope scans [1] and line responses from linear shift-invariant optical systems.

Because these algorithms will be implemented on a digital computer, we are constrained to work with sampled data. Our input projections must be sampled and our output reconstruction will consist of samples of the unknown structure. As a result this problem is best considered as an inherently digital one. This constrains us somewhat by limiting the class of signals that can be reconstructed, but this is not a serious concern since most signals that arise in practice can be closely approximated by signals from this restricted class. On the other hand, by constraining ourselves to this one class of signals, we have developed some extremely powerful algorithms and interesting results.

THE GENERAL RECONSTRUCTION ALGORITHM

Up to this point we have assumed that projection functions are simply the mathematical equivalents of x-ray photographs or electron micrographs. To understand the problem more fully we must make that definition more precise. Let us assume that one unknown signal can be described by an extinction function $f(x,y,z)$ and that an x-ray photograph is made of the unknown by a uniform, collimated x-ray beam with intensity I_0 which propagates parallel to the y-axis. Then, ignoring scattering effects, the observed intensity variation of the x-ray photograph can be described by:

$$I(x,z) = I_0 \exp\left\{ -\int_{-\infty}^{\infty} f(x,y,z)\, dy \right\} \qquad (1)$$

We can then define the projection function associated with this orientation by

$$p_0(x,z) = -\ln\left[\frac{I(x,z)}{I_0}\right] = \int_{-\infty}^{\infty} f(x,y,z)\,dy \qquad (2)$$

The function of $f(x,y,z)$ might measure the density of the unknown as it does to some extent in the case of x-rays or it might measure the extent of staining of a specimen in the case of electron micrographs, etc.

To generalize eq. (2), let us assume that each projection is taken normal to the z-axis and that the projection plane, the u-z plane, meets the x-z plane at an angle θ. With this notation, eq. (2) corresponds to the projection at angle $\theta=0^{\circ}$. We shall define the projection at angle θ by

$$p_\theta(u,z) = \int_{-\infty}^{\infty} f(u\cos\theta - v\sin\theta,\ u\sin\theta + v\cos\theta,\ z)\,dv \qquad (3)$$

If $F(\omega_x,\omega_y,\omega_z)$ is the Fourier transform of $f(x,y,z)$, and if we define a plane, the ω-ω_z plane to intersect the ω_x-ω_z plane at an angle ϕ, then we can further define the slice of $F(\omega_x,\omega_y,\omega_z)$ at angle ϕ to be $F(\omega\cos\phi,\omega\sin\phi,\omega_z)$, i.e., a slice of $F(\omega_x,\omega_y,\omega_z)$ corresponds to the function evaluated on a plane, which includes the ω_z axis, and which is specified by a single angular parameter. These relationships are illustrated in Fig. 1.

Theorem: (projection/slice theorem) The projection of $f(x,y,z)$ at angle θ to the x-z plane is the Fourier transform of the slice of $F(\omega_x,\omega_y,\omega_z)$ at angle θ to the ω_x-ω_z plane.

Proof: Let us define a coordinate system, the $\omega,\hat{\omega},\omega_z$ coordinate system, which is a rotation by θ (radians) about the ω_z-axis, of the $\omega_x,\omega_y,\omega_z$ coordinate system.

$$\begin{aligned} \omega &= \omega_x\cos\theta + \omega_y\sin\theta \\ \hat{\omega} &= \omega_x\sin\theta + \omega_y\cos\theta \end{aligned} \qquad (4)$$

Then since

$$F(\omega_x,\omega_y,\omega_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)\exp(-j[x\omega_x+y\omega_y+z\omega_z])\,dxdydz \qquad (5)$$

we can write

$$\hat{F}(\omega,\hat{\omega},\omega_z) = F(\omega_x,\omega_y,\omega_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)\exp(-j[x\omega\cos\theta + y\omega\sin\theta - x\hat{\omega}\sin\theta + y\hat{\omega}\cos\theta + z\omega_z)]\,dxdydz\,. \qquad (6)$$

Since

$$F(\omega\cos\theta,\omega\sin\theta,\omega_z) = \hat{F}(\omega,0,\omega_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)\exp(-j[x\omega\cos\theta + y\omega\sin\theta + z\omega_z])\,dxdydz \qquad (7)$$

by defining

$$\begin{aligned} u &= x\cos\theta + y\sin\theta \\ v &= -x\sin\theta + y\cos\theta \end{aligned} \qquad (8)$$

we can observe

$$F(\omega\cos\theta,\omega\sin\theta,\omega_z) = \hat{F}(\omega,0,\omega_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left\{\int_{-\infty}^{\infty} f(u\cos\theta-v\sin\theta,\ u\sin\theta+v\cos\theta,\ z)\,dv\right\}\exp\{-j(u\omega+z\omega_z)\}\,dudz \qquad (9)$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_\theta(u,z)\exp\{-j(u\omega+z\omega_z)\}\,dudz \qquad (10)$$

Q.E.D

This theorem forms the basis for our reconstruction algorithms. First we obtain as an input a set of projections (say from processed x-rays). It is most convenient if these are all taken normal to a single axis which we have defined as the z-axis in our coordinate system, although this is not necessary for the projection/slice theorem can be generalized. Then we transform these projections, by low pass filtering them, sampling them and performing a discrete Fourier transform (DFT) calculation. These samples of the Fourier transforms of the set of projections provide us with samples of $F(\omega_x,\omega_y,\omega_z)$ by virtue of the projection/slice theorem. From these we can estimate the entire Fourier space and then by inverse Fourier transforming in three dimensions, we have an estimate of $f(x,y,z)$, or, in our terminology, a reconstruction. How many projections are needed, how they should be sampled and how the Fourier space should be estimated from samples of the slices depend upon the individual algorithms.

SPECIFIC RECONSTRUCTION ALGORITHMS

Let us assume that the functions to be reconstructed are bandlimited. This assumption will guarantee that all of the projections are bandlimited and it will thus allow us to compute Fourier transforms from samples of the projections with no loss of information, a necessity for performing a reconstruction digitally. Also for our arguments let us assume that we are trying to reconstruct a two-dimensional function (picture) from one-dimensional projections. This is equivalent to reconstructing for only a single value of z.

One approach to use to obtain samples of the Fourier transform of an unknown picture is to sample each projection at the same sampling rate, that rate being greater than the maximum Nyquist rate of each of the projections. If then each sequence of samples is Fourier transformed using a DFT algorithm, the Fourier transform of the unknown picture $f(x,y)$ will be known on a polar raster of points. If the projections were evenly spaced from 0 to π, the raster of points is that shown in Fig. 2(a). From these samples of $F(\omega_x,\omega_y)$ there are a number of techniques that can be used to estimate $f(x,y)$. One of the more successful that we have used is to use linear interpolation to estimate the values of $F(\omega_x,\omega_y)$ on a Cartesian (square) raster from the polar samples. From the Cartesian raster we can perform an inverse two-dimensional DFT to obtain our estimate. Another approach that has yielded good reconstructions from a polar raster is to write the inverse transform integral in polar coordinates,

$$F(x,y) = \frac{1}{4\pi^2} \int_0^{\infty} \int_{-\pi}^{\pi} \hat{F}(\omega,\theta) \exp\{j(x\omega\cos\theta + y\omega\sin\theta)\}\omega \, d\omega d\theta \qquad (11)$$

and then approximate (11) by a sum, where each summand depends upon one of the polar samples. The latter technique performs well resolving detail but does poorly on areas of low information content, such as backgrounds or areas of nearly constant grey level. The interpolation technique, on the other hand, does just the opposite. It resolves the flat areas but does not extract detail well.

As an alternative to sampling each projection at the same sampling rate we can vary the sampling rate with the projection angle. In particular, assume that $f(x,y)$ is bandlimited within a square in the frequency domain such that $F(\omega_x,\omega_y) \equiv 0$ for $|\omega_x| \geq W$ or $|\omega_y| \geq W$. Then the bandwidth of the projection at angle θ is defined by

$$W_\theta = \frac{W}{\max\{|\cos\theta|, |\sin\theta|\}} \qquad (12)$$

Now suppose we sample each projection at a sampling rate which is proportional to its bandwidth. If these sequences are then DFT'd, $F(\omega_x,\omega_y)$ will be known on a concentric square raster such as that in Fig. 2(b). At first glance such a set of points is not to be favored over the polar set, but such is not the case. Using both the interpolation algorithm and the integral approximation algorithm, we get better reconstructions from the concentric squares raster than from the polar one. One reason for this can be seen from the fact that the Cartesian raster of samples which we must have in order to perform an inverse DFT lie along the same horizontal and vertical lines as the sides of the concentric squares. Thus instead of the necessity of performing a two-dimensional linear interpolation, we only need to perform a one-dimensional one. This might be expected to produce less error.

The second advantage to a concentric squares raster becomes apparent when we consider a special class of signals. We have assumed that $f(x,y)$ is bandlimited. Let us now assume that it can be represented in the form

$$f(x,y) = \sum_{m=C}^{N-1}\sum_{n=0}^{N-1} f\left(\frac{m\pi}{W},\frac{n\pi}{W}\right) \frac{\sin W(x-\frac{m\pi}{W})\,\sin W(y-\frac{n\pi}{W})}{W^2(x-\frac{m\pi}{W})(y-\frac{n\pi}{W})} \tag{13}$$

Thus in addition to requiring that $f(x,y)$ be bandlimited, we have required that when sampled at its Nyquist rate it have only a finite number of non-zero samples. The number N which represents the width of the square of non-zero samples we will refer to as the order of $f(x,y)$. Bandlimited functions of finite order are completely specified by their DFT's and their Fourier transforms are two-dimensional polynomials of degree N-1 in each variable. Because of the fact that a one-dimensional polynomial of degree d is completely specified by d+1 independent samples, it can be proven that a bandlimited function of order N can be reconstructed exactly from N+1 projections (although not necessarily for all sets of N+1 projections). Thus for this class of functions, concentric square projections as these projections shall be called, assume theoretical importance.

In Fig. 3 is shown a reconstruction of a cross section of a leg bone from 36 concentric squares projections. The projections were sections of x-rays which were taken at 5^0 intervals which are scanned logarithmically. Each section of each x-ray consisted of 256 samples and the reconstruction is plotted on a 256x256 Cartesian raster.

RECONSTRUCTING FROM ONE PROJECTION

In the previous section we noticed that bandlimited functions of finite order could be reconstructed exactly from N+1 (concentric squares) projections, where N was the order of the function. Is this the minimum number of projections needed to reconstruct these functions? The answer to this question is no. In fact, functions of this form can be reconstructed from a single projection.

Theorem: (one projection theorem) A bandlimited function $f(x,y)$ of order N in each variable can be reconstructed from the concentric squares projection at $\theta = \tan^{-1}(1/N)$.

Proof: Consider the slice at $\theta = \tan^{-1}(1/N)$.

$$F(\omega\cos\theta,\omega\sin\theta) = F\left(\frac{\omega N}{\sqrt{N^2+1}},\frac{\omega}{\sqrt{N^2+1}}\right) \tag{14}$$

where

$$F(\omega_x,\omega_y) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(\frac{m\pi}{W}, \frac{n\pi}{W}) \exp(-j[\frac{\pi}{W}(m\omega_x+n\omega_y)]) \; b_{WW}(\omega_x,\omega_y) \qquad (15)$$

where

$$b_{WW}(\omega_x,\omega_y) = \begin{cases} 1, & |\omega_x| \le W, \; |\omega_y| \le W \\ 0, & \text{otherwise} \end{cases}$$

Evaluating (14),

$$F(\frac{\omega N}{\sqrt{N^2+1}}, \frac{\omega}{\sqrt{N^2+1}}) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(\frac{m\pi}{W}, \frac{n\pi}{W}) \exp\{-j(\frac{\pi\omega}{W\sqrt{N^2+1}}[Nm+n])\} \qquad (16)$$

$$|\omega| \le \frac{W}{N}\sqrt{N^2+1}$$

If we define $g(Nm+n) = f(\frac{m\pi}{W}, \frac{n\pi}{W})$, then (16) becomes

$$F(\frac{\omega N}{\sqrt{N^2+1}}, \frac{\omega}{\sqrt{N^2+1}}) = \sum_{p=0}^{N-1} g(p) \exp(-j\frac{\pi\omega p}{W\sqrt{N^2+1}}), \quad |\omega| \le \frac{W}{N}\sqrt{N^2+1}$$

$$= 0 \qquad \text{otherwise} \qquad (17)$$

Thus over the region of interest, the slice at $\theta = \tan^{-1}(1/N)$ (which can be obtained from one projection) is a one-dimensional polynomial of degree N^2-1 in the variable $\exp\{-j(\pi\omega/W\sqrt{N^2+1})\}$, and the coefficients of that polynomial are simply the picture samples arranged column by column. Thus knowledge of N^2 sample values specifies the picture samples and by (13) this specifies the unknown picture function.

Q.E.D.

This theorem also has implications in two-dimensional filter design for it implies that the frequency response of a two-dimensional non-recursive digital filter is specified by its values along a single radial line, although how this fact might be utilized in filter design is still not clearly understood.

Despite the beauty of the one-projection theorem, it is not particularly useful as a reconstruction technique on a finite precision machine, because for values of N larger than 4 or 5, solving eq. (17) for $\{f(m\pi/W, n\pi/W)\}$ requires the inversion of a large nearly singular matrix and any errors made in obtaining the slice samples, which are bound to occur, are amplified enormously. In this respect, applying the one projection theorem is much like trying to apply analytic continuation to an unknown function all of whose derivations are known at a single point.

From a theoretical viewpoint, it is interesting to ask if there are any other of these critical slices. Clearly not all slices will work; for example, knowledge of sample values along just the ω_x or ω_y axis is generally not sufficient. In fact, there are an infinite number of these critical slices. A necessary and sufficient condition for a slice with a rational slope to be a critical slice is given in the following theorem which will be stated without proof.

Theorem: If a slice has an angle $\tan^{-1}(A/B)$ where A and B are integers with no common factor, then a necessary and sufficient condition for this slice to be sufficient for reconstruction of a bandlimited function of order N is that the equation

$$Bm + An = Bm' + An', \quad 0 \le m,n,m',n' \le N-1$$

possess only the trivial solution $m = m'$
$n = n'$.

In particular, if N is a power of two and A is an odd integer, the slice with slope $\tan^{-1}(A/N)$ is critical.

Using techniques completely analogous to that by which we derived the one projection theorem, we can derive two projection theorems, four projection theorems, etc. As a rule of thumb, the greater the number of projections which we care to use, the easier it is to obtain a solution with real data. It generally takes about N/2 projections for the sensitivity to be reduced enough to use currently available machines.

SUMMARY

A number of algorithms have been presented for reconstructing multidimensional signals from their projections, such as x-rays. In addition to their uses in performing reconstructions, projection functions are potentially useful for characterizing multidimensional signals for purposes of pattern recognition or bandwidth compression for signal transmission. Uses for projections in these areas has not been explored.

ACKNOWLEDGEMENT

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense, monitored by ONR under Contract No. N0014-67-A-0204-0064, and by National Science Foundation Grant No. GK-31353.

REFERENCES

1. R.N. Bracewell, "Strip Integration in Radio Astronomy", Austr J. Phys., 9:198, 1956.

2. R.A. Crowther, D.J. DeRosier and A. Klug, "The Reconstruction of a Three-Dimensional Structure from Projections and its Application to Electron Microscopy", Proc. Roy. Soc. (Lond.) A, 317:319, 1970.

3. D.J. DeRosier and A. Klug, "Reconstruction of Three-Dimensional Structures from Electron Micrographs", Nature, 217:130, 1968.

4. J.B. Garrison, D.G. Grant, W.H. Guier and R.J. Johns, "Three-Dimensional Roentgenography", Am. J. Roentgenology, Radium Therapy and Nuclear Medicine, 60:903, 1969.

5. R. Gordon, R. Bender and G.T. Herman, "Algebraic Reconstruction Techniques (ART) for Three-Dimensional Electron Microscopy and X-Ray Photography", J. Theo. Biol., 29:471, 1970.

6. R.M. Mersereau, "Recent Advances in the Theory of Reconstructing Multi-Dimensional Signals from Projections". MIT Res. Lab. Electron. Quart. Progr. Rept. No. 105:169, 1972.

7. G.N. Ramachandran and A.V. Lakshminarayanan, "Three-Dimensional Reconstruction from Radiographs and Electron Micrographs III: Description and Application of the Convolution Method", Indian J. Pure Appl. Phys. 9:997, 1971.

8. O.J. Tretiak, M. Eden and W. Simon, "Internal Structure from X-Ray Images", 8th ICMBE, 1969.

9. B.K. Vainshtein, "Finding the Structure of Objects from Projections", Sov. Phys. Crys., 15:781, 1971.

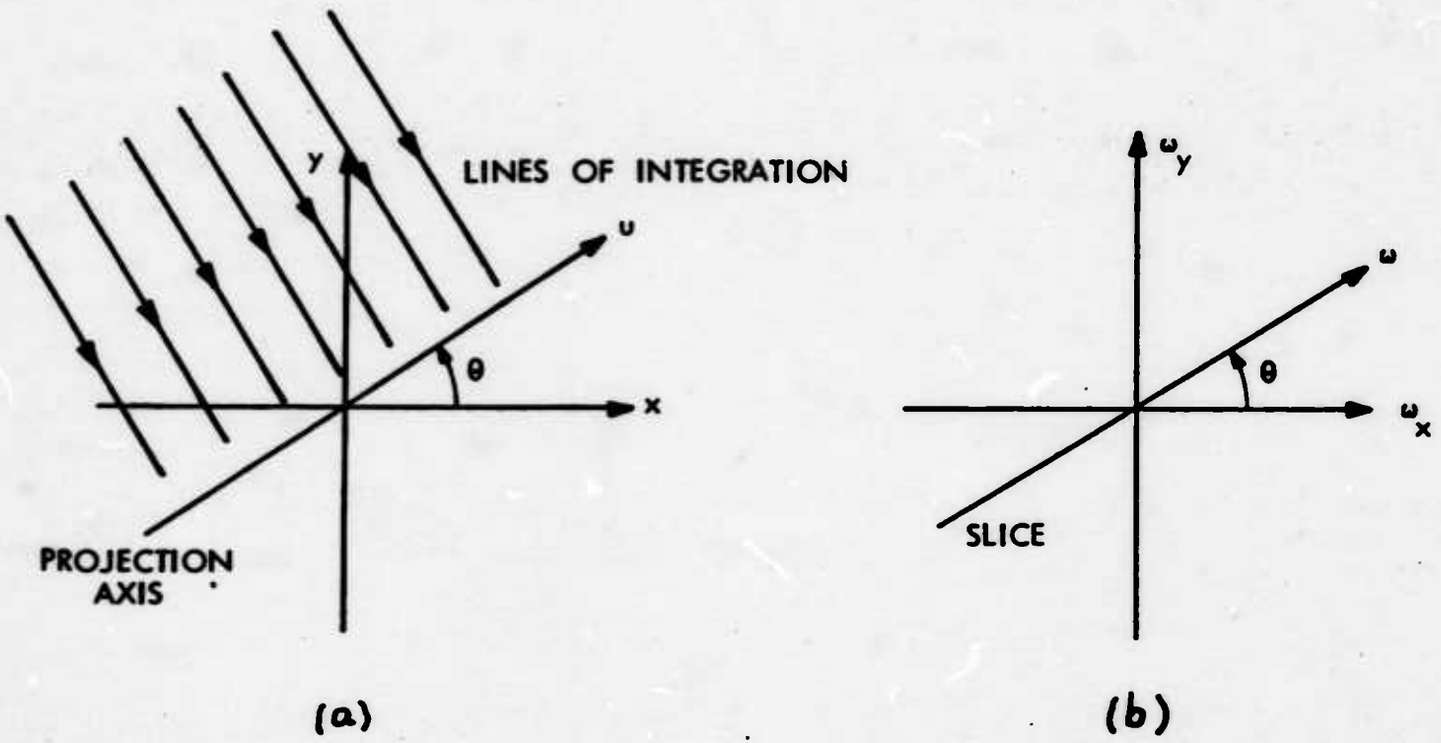
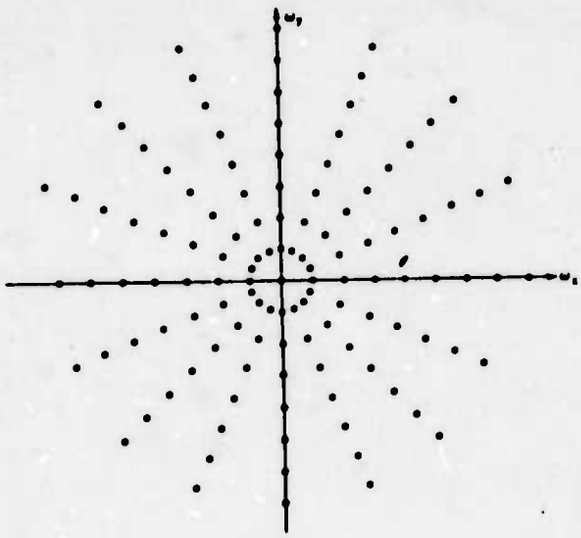
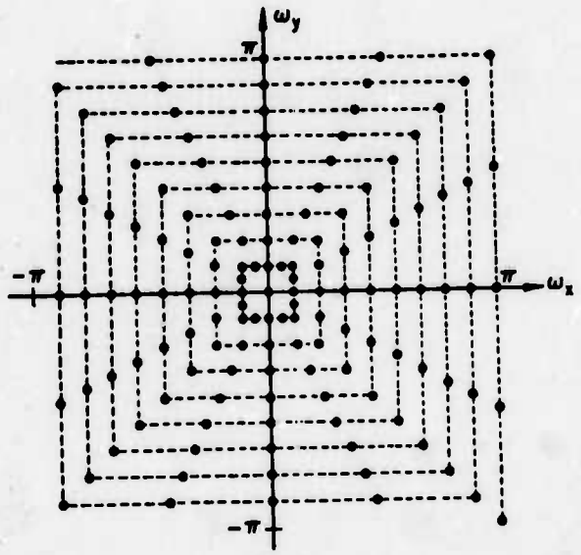


Figure 1 — Definitions of (a) the projection of $f(x,y,z)$ at angle θ. The z-axis is normal to the page. (b) The slice of $F(\omega_x,\omega_y,\omega_z)$ at angle θ. The ω_z-axis is normal to the page.



(a)



(b)

Figure 2 — The samples of $F(\omega_x,\omega_y)$ that can be obtained by (a) sampling 8 projections all at the same rate, (b) sampling each of 8 projections at a rate proportional to its bandwidth.

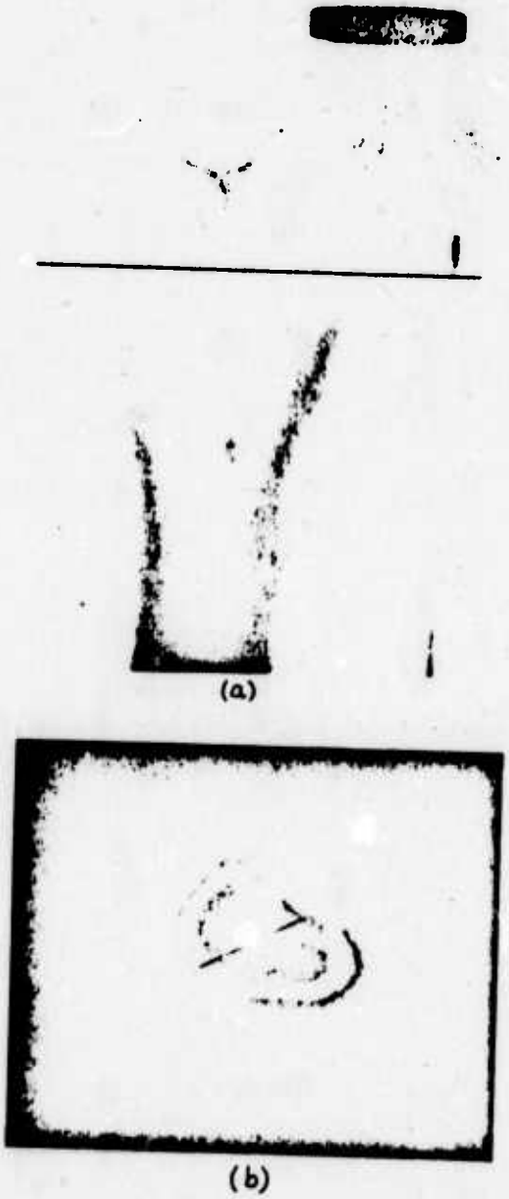


Figure 3 — A reconstruction of a section of a human femur from 36 concentric squares projections at equally spaced angles. (a) One of the x-rays from which the reconstruction was made. (b) The reconstructed cross-section. Note: The two photographs are not to scale. The photgraph in (a) is a positive, that in (b) is a negative.

THE BLACK BOX

being a

small, fast, inexpensive

digital processor

designed mainly for

speech synthesis

but nicely suited

for

myriad other tasks

J. Allen

F.X. Carroll

E. Jensen

This document contains a detailed description of the proposed Black Box. Readers are asked to comment freely and quickly so we can proceed with construction.

The Black Box

1. Introduction. In this paper, we will describe a new computer having several unusual design features. The original motivation for this design was the need for a real-time all-digital speech synthesizer. Since the vocal tract model to be simulated is complicated (see Fig. 1, due to D.H. Klatt), it was necessary to adopt several features which optimize these calculations in time. The heart of the computer is a very fast multiplier (18 X 18 in about 100 nsec). No instruction overlap is used, but instructions have a three-address format, so that (for example) A-B → C is done in one instruction, including the two loads and one deposit. Each instruction contains an op-code plus these three addresses, as well as a jump address for the next instruction, so that no program counter is needed. There is a separate instruction memory of 44-bit width, and two data memories, each of 18-bit width. It should be noted that no special registers are provided. There is no accumulator, no program counter, and the machine status word, as well as the direct memory access word count and addresses are kept in memory, so that they can always be inspected by the program. Very little timing is needed internal to an instruction, since the multiplier is combinatorial, and shifting is accomplished via multiplexing. This results in a fairly simple control structure, most of the complication arising from I/O considerations. The computer is designed to be interfaced to a host PDP-9 machine, and will probably not be used in a stand-alone mode, although this is possible. Direct memory access transfers are possible in either direction between the PDP-9 and any memory location in the black box at a 1 megacycle rate. In addition, transfers to and from the PDP-9 accumulator and any black box location are possible. PDP-9 IOT instructions can be used to control various control bits in the black box. Finally, part of the data memory is paralleled, so that it is possible to compute using one parameter set, while a new set is being loaded (transparently to the black box program) from the PDP-9. In this paper,

we present several examples to show the utility of this device for other specialized tasks, including display processing, floating point calculations, and FFT's. The basic design goal has been to achieve a powerful, fast, yet inexpensive processor, but with little consideration given to ease of programming.

2. Architecture. Figure 2 shows a rough block diagram of the computer. Three components can be recognized.

1) Instruction processor: Includes a 256 X 44-bit instruction memory, loadable directly from the host machine; an instruction register; instruction addressing; skip logic; and instruction decoding.

2) Data processor: Includes

a) Two data memories, X + Y, each 256 X 18, and each having some of its locations paralled by additional memory locations.

b) Three processing units:

1. Multiply unit, performs X.Y+Z
2. Arithmetic-Logical Unit (ALU), performs adds, subtracts, and logical operations.
3. BIT test, performs skip on a selected bit of a given word and provides for modification of that bit.

c) Data Select: Selects desired output from processing unit, with capability to incorporate shifts.

3) Input-Output Processor; provides programmed and DMA transfers with the host computer.

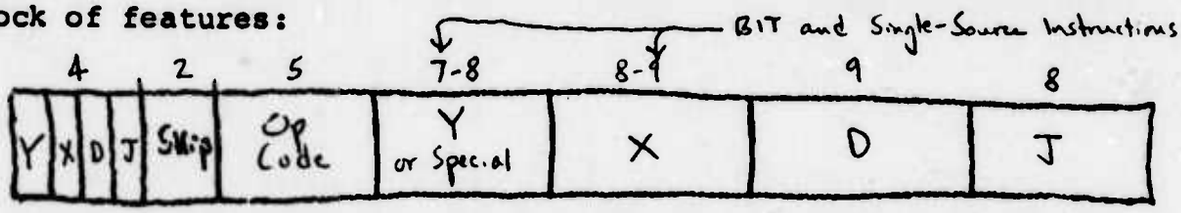
Figure 3 shows a more detailed block diagram, which is intended to relate to the further detailed discussion.

3. Instruction processor: Instructions are 44 bits long, and cannot be modified within the black box except that the effective address of all address fields can be modified by index registers, or an instruction may be skipped. The host computer, however, has access to the 256-location instruction memory so that instructions may be updated at any time

by input transfers into the black box. There is no program counter, so that each instruction contains a jump address to the next instruction. This jump address can be modified by indexing or by ORing into its LSB the OR of the following 3 conditions:

1) Data Select output = zero
2) " " " negative
3) one bit selected by BIT instruction.

There are five classes of instructions. All five have a common block of features:



The only variations in this format over the entire group of instructions are the length of the X-address field (8 or 9 bits) and the specifics of the "Y or special" field, which is either the Y-address (in double-source instructions) or the specifier of actions peculiar to certain single-source instructions. We first describe the common features (i.e. everything but the "Y or special" field):

1) Index control: There are 4 index registers located in the X memory

X-Address	Register	Action
11_8	X_Y	modifies Y-address
12_8	X_X	" X- "
13_8	X_D	" D- "
14_8	X_J	" Jump "

The 4 index bits control the ORing of these registers into their respective address registers. The use of ORing, as opposed to adding, simplifies the logic and increases the speed of indexing at very little cost in programming convenience. In single-argument and BIT instructions, the Y index bit is ignored.

2) Skip bits: There are always two possible skips associated with each instruction.

N skip on data select output negative
Z skip " " " " = zero

Placing 1's in these fields enables the skips, which cause a 1 to be ORed into the jump address LSB if the designated skip condition is met.

3) Op Code: The op code is 5 bits long. Detailed explanations are given below.

4) X-address. In double-source instructions, this is an 8-bit address in the X-memory. In single-source instructions, the 9 bits designate a location in X or Y memory. If the MSB is 0, the X-memory is addressed, and if it is a 1, the Y-memory is addressed.

5) D-address: This is the 9 bit destination address, i.e. where the result of an instruction is stored. When the memory is in the serial mode, the result can be stored in any X- or Y-location, but when the memory is in parallel mode, the result is stored in the analagcus locations, specified by the 8 least significant bits, in both memories. This last situation is violated in input transfers, discussed below.

6) Jump Address: 8-bit address of next instruction.

This block of features is augmented in different ways to form the five instruction classes:

I) Double-Source Instructions: The X and Y fields are both 8 bits long, and each specify a source location in their respective memories. See Figure 4. The result is stored in D, and the next instruction taken from J.

II) Multiply: X and Y are 8-bit source addresses, and a third source is always implied, namely the Z-register, which

is location 10_8 in the X-memory. See Figure 5. Note that Z is always cleared before a deposit is made, so that it will either be properly updated when D = Z, or set to zero to provide only the X·Y function. The op-code contains a 3-bit scaling field, and the 8 possible options are shown in Figure 5.

III) Index Register Modification Instructions: These look like an ADD instruction in that the contents of the two source addresses are added and stored in the destination address, except that the result is also stored in the index register referred to by the instruction. See Figure 6. There are 4 index instructions, one for each index register.

IV) BIT instruction: The X-field is 9 bits long so that any X or Y location can be used as source. The remaining (special) field of 7 bits is divided into a 5-bit select field and a 2-bit modification field. See Figure 7. The select field specifies which of the 18 bits of a data word is to be examined. If this bit is a 1, it is ORed into the LSB of the jump address. The modification field can then affect this bit in any of 4 ways, shown in the figure. Additionally, after this modification is made, the usual skip tests can be performed on the resultant data word.

V) Single-Source Instructions: Here, the X-address field is 9 bits long, and the remaining 7 bits is used to specify shifts and rotates as shown in Figure 8. Thus both source and destination can be any address in X or Y.

The instruction processor has the following paths to the rest of the machine:

a) Index bits: cause index registers to be ORed into the corresponding address registers

b) Skip bits: cause outputs from skip tests to be ORed into the jump address

c) Op code including "special" field when present: goes to instruction decoder, then to instruction execution control in the data processor.

d) Source and destination addresses: routed to respective data memory address registers.

e) Jump address: routed to instruction address register.

4. Addressing Figure 9 shows the complete address space, both as seen internally and by the host computer. Note that

1) To the host machine

a) the X + Y memories are one 512-register memory, locations 0-777

b) each instruction memory location is mapped onto 4 host computer 18-bit words, so that 4X256 = 1024 host memory words are needed to represent the instruction memory. These are locations 2000-3777 of the overall address space.

2) The top 40_8 locations of the Y memory can be switched between two separate physical memories. More on this below.

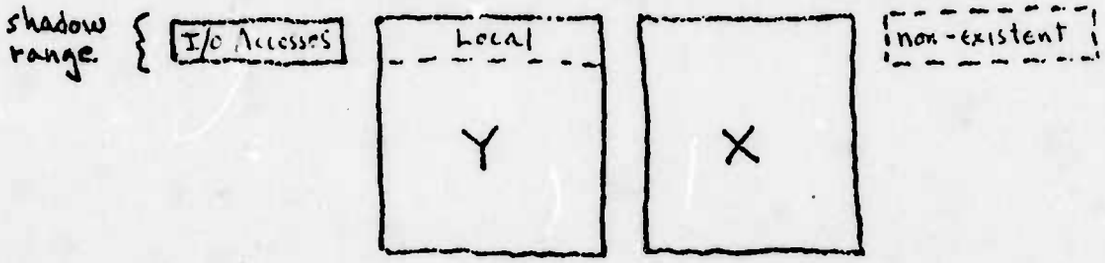
3) The bottom 15_8 locations of the X memory are special registers, which can be accessed directly in addition to the normal X-addressing.

5. Data Processor: As mentioned above, there are 3 units in the data processor:

a) X + Y data memories: These are 256 X 18-bit memories, which can run in one of 2 modes selected by a control bit in Word 0 of the X-memory.

1) Serial: This mode logically places the X and Y memories end-to-end. X is the bottom 256 locations, and Y the top 256 locations. Thus, in this mode, all deposits and single-source-instruction loads have access to all 512 locations. Additionally, the I/O instruction (which is a single-source instruction) can access all 512 locations in this mode. Of course, double-source instructions access both X + Y separately and simultaneously.

2) Parallel: Source loads are unaffected by this mode, but deposits go to analogous locations in both X + Y memories. In the following diagram, the case of the shadow memory in parallel mode is shown. When the shadow mode and parallel mode are both selected, input and output transfers within the shadow memory range go to/from that memory which is selected



Logical (not physical) arrangement of data memories

as the I/O shadow memory at the time of the transfer, and no-where else. It is as though the other part of the parallel transfer went to a (non-existent) X shadow. Thus, in general, even in the parallel mode, the contents of X & Y within the shadow range will not

agree, and this is desirable. Switching of the shadow memories only effects which of the two is currently local to the black box Y memory, and which is used for I/O. (See below)

The X memory has its bottom 15_8 registers used for special purposes, as shown in Figure 10. These registers can be used just like any other memory register, but in addition, registers 4-14 have special direct output lines, and registers 0-3 have direct input and output lines to the host computer. Additionally, locations 1, 2, 3, and 7 are counters. All of these special registers are treated in detail below. Word 0 is the control register, and is also given special treatment.

The Y memory has its top 32_{10} registers duplicated by a "shadow" memory. (See Figure 3) When the shadow mode is off, the Y memory is a straightforward 256 X 18 memory. But when the shadow mode is on, a bit in the control register selects which of the two memories will accommodate local loads and deposits within this address range. I/O with the host computer, however, will utilize that memory not so selected, and this is done automatically. This feature was provided to allow for a parameter memory, one part of which could be used for local computing, while the other is being updated by the host computer.

b) Processing Units: Within the data processor, there are three processing units, specialized by function.

1) Multiply As described in Figure 5, this unit performs X·Y+Z. Z is supplied by register 10 of the X memory, and is always used in the calculation. An 18X18 array multiplier perform full 2's complement multiplication, together with the Z add, in about 100 nsec. All scalings are done in the data select unit.

2) Arithmetic-Logical Unit (ALU) This is a processor accepting X & Y inputs and performing adds, subtracts, and logical operations. It is realized in 74S181's and is used for all instructions except multiply.

3) Bit test (BIT) This unit selects one among the 18 bits of a data word, and connects it to the jump address skip logic. Thus, the selected bit is Ored into the LSB of the jump address. The selected bit can also be modified, as shown in Figure 7, and this is accomplished in the ALU.

c) Data Select: This unit is a large multiplexor, capable of gating any one of [illegible] 18-bit words through the unit. These words include input transfers, multiplier scalings, and ALU shifts and rotates. Note that all shifting is done by multiplexing, so that no shift register or counter is required.

Skip tests for negative and zero are made at the output of the data selector, and all results are held in a latch, while the destination address is set up, before they are stored.

6. Input/Output Processor: This processor handles all I/O transfers with the host computer. As shown in Figure 9, the host machine has access to all memory locations in the black box. The I/O processor accomplishes each transfer by executing a one-instruction interrupt (see below), which is transparent to the currently running program. This is the only interrupt facility in the black box. Transfers may take place in either direction, and are of two types.

a) AC transfers. When the PDP-9 is the host, transfers to and from its AC can be made under host I/O control. Thus these transfers can only be originated from the host machine.

b) DMA transfers. DMA transfers (up to a 5 megacycle rate) can be initiated by either machine. The required special registers are in the X-memory.

X-Location	Use
1	Host DMA address
2	Local DMA address
3	Word count

Note that all three of these registers are counters. The details of these transfers are as follows: (IOT refers to host I/O control instructions).

a) AC transfers (LOC refers to an arbitrary black box data location)

1. PDP-9 AC → BB LOC
   a) IOT puts BB address into BB X_2, via input buffer register (IBR)
   b) IOT puts data in IBR, and causes interrupt to transfer data to LOC
2. BB LOC → PDP-9 AC
   a) IOT puts BB address into BB X_2, and then causes $C(X_2)$ to be loaded into Output Buffer Register (OBR)
   b) IOT calls for data from OBR → AC

b) DMA transfers

1. PDP-9 DMA to BB
   a) IOT host starting address to X_1
   b) IOT black box starting address to X_2
   c) IOT word count to X_3
   d) IOT to initiate transfer: direction of transfer and shadow mode control are also specified.
2. BB to PDP-9 DMA
   This is similar to the above, except for direction of transfer.
3. DMA transfers can also be initiated directly within the black box, since all those special registers, and the DMA control flags are within the X-memory. Obviously, these registers can be inspected at any time.

The I/O instruction, executed during the I/O interrupt, is a somewhat specialized MOVE (See Figure 8). Thus it has 9-bit source and destination addresses.

a) Input LOC is in the DMA BB Address register (X memory word 2); data select from Input Buffer register is enabled, and a MOVE takes place from LOC to LOC. Thus the initial contents of LOC are placed on the X-bus, but the outputs of the data processors are not gated through data select. In this way, the initial contents of LOC are lost, and the Input transfer only uses the "store" part of the MOVE instruction.

b) output LOC is in the DMA BB Address register. A MOVE from LOC to LOC is executed, and the output of the data selector (the contents of LOC) is also placed in the Output Buffer Register (OBR).

7. Special Registers. As previously mentioned, the bottom 15_8 locations of the X data memory are special, in that they are more than plain memory locations. Some have special input and/or output lines, and may even be counters. The several subsets of these registers require special discussion.

a) $11\text{-}14_8$. These are the 4 index registers. The index instructions (Figure 6) store into them as well as the designated destination address. The outputs of these registers are ORed into their respective address fields if gated by 1's in the respective index control bits of the instruction being executed. These registers are not counters. All index incrementing must be performed explicitly.

b) 10_8. This is the Z-register. Its output lines are permanently connected to the multiplier, so that a multiply instruction always adds the contents of Z to X·Y to form the output of the multiplier.

c) 6-7 These are the Clock and Count registers,

respectively. Normally the clock interval is in the clock register. This count is automatically jammed into the count register when the latter goes to 0 from -1. This condition is equivalent to the transition to 0 being coupled with a carry out of the MSB. Thus no jamming takes place when 0 is deposited into the count register. Each time the desired interval is jammed into the count register, a pulse is generated for external use. The count register is also constructed so that it cannot count beyond 0. If a periodic output pulse is desired, merely set the desired period in the clock register, and the count register will generate output pulses at the period interval repeatedly. An external clock supplies the pulse train for the count register. Any existing count can be overridden at any time by simple depositing into the count register. This scheme, of course, also provides for changes from one period to another, by merely updating the clock register at the appropriate time, i.e. before the current period expires. Also, it is clear that no special instructions are needed for these operations.

d) 4-5 These are the Output Registers. Their outputs are brought directly (buffered, of course) to rear-panel connectors. The intent is to connect D/A converts to these outputs.

e) 1-3 These are the DMA address and word registers. Location 1 contains the host computer current address, location 2 the local (black box) current address, and location 3 the present word count. This last location generates a pulse on the transition from -1 to 0 for use in supplying an interrupt to the host computer. All three registers are counters.

f) 0 This is the Control Register. Its individual bits are employed as follows: (no significance is attached to the order.)

1. (1 bit) Run/Halt. DMA will still transfer during Halt.
2. (1) Link. This flag is set on carry out of the MSB during ADD or SUB. There are corresponding instructions which provide ADD with Link and SUB with Link.
3. (1) Overflow. This flag is set on overflow during ADD and SUB (and possibly on some multiplier outputs). It can be reset only by program control (using BIT).
4. (2) Shadow. One bit turns the shadow mode on, so that I/O transfers use the memory not selected, and the other bit selects which physical shadow memory is used by the BB for local computation.
5. (1) Series/Parallel. In the serial mode, deposits are made to one of 512 locations in X or Y, whereas in parallel mode deposits go to two locations, one in each of X and Y. (Note comments above concerning I/O in parallel mode within the shadow address range).
6. (3) DMA There are 3 control bits
   a) Start. Resets when transfer is done, and initiates interrupt (when enabled) to the host computer when done.
   b) Direction. Into or out of the Black Box.
   c) Interrupt Enable. Enable flag allows Black Box to interrupt host computer when DMA transfer is done.
7. (1) Clock: This is an enable flag, which allows the counter-register → 0 condition to interrupt the host computer.

8. (1) Program Interrupt. This is a general flag which can be used to interrupt the host computer for any reason.

9. (1) Program Interrupt Enable. Provides overall control of all interrupts from the Black Box to the host computer.

Items 1-9 above account for 12 bits, so there are still 6 bits left which can be used for flags or other purposes.

8. Front panel: The front panel will have two sets of switches and corresponding lights. One set is 18 bits long for data or instructions. The other set is 12 bits long and specifies the address for a location. Coupled with DEPOSIT and EXAMINE keys, it is thus possible to examine and change any location in any memory while the computer is halted. This process will, however, destroy the previous contents of register 2 of the X-memory (Black Box DMA address), since this location is used to specify the black box address of all I/O transfers.

In addition to the DEPOSIT and EXAMINE keys, START, STOP, CONTINUE, and SINGLE STEP will be provided, as well as EXAMINE NEXT and DEPOSIT NEXT. The address for the first instruction to be executed following START is located in register 2 of the X-memory.

Additional display lights will include the Instruction Register (IR), Input Buffer Register (IBR), and Output Buffer Register (OBR).

9. Miscellaneous

a) Timing: It is anticipated that all instructions except multiply will take 200 nsec, and multiply will require 300 nsec. These are conservative figures and, as more packages become available in Schottky TTL, will probably be revised downward.

b) Size: We expect that the entire processor will require 5 1/4" rack height. About 300-400 IC's will be required (the multiplier alone requires 100 74S181's).

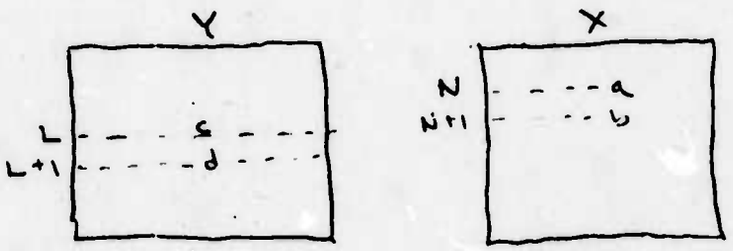
10. Programming Examples: In this section we show several programs, ranging from the straightforward to the spectacularly obscure, which illustrate design features of the machine.

a. Complex multiplication:

$$\frac{a+jb}{c+jd}$$

$$(ac-bd)+j(ad+bc)$$
$$S \quad T \qquad u \quad v$$

Initially, let X and Y index registers contain N and L respectively, and the D index register contain M. The result is to go in locations M and M+1. Therefore, initially, we have;



Note: in parallel mode, the final result (M+M+1) will be in both X + Y memories

```
CMULT   MUL   *, *, S      / (ca) → S
        MUL   *1, *1, T    / (bd) → T
        MUL   *, *1, u     / (ad) → u
        MUL   *1, *, v     / (bc) → v
        SUB   S, T, *      / (ac-bd) → M
        ADD   u, v, *1     / (ad+bc) → M+1
```

Note: "*" means enable index register for indicated field. "*1" 1 ORed with the appropriate index register.

b. Second Order Difference Equation: The second order difference equation,

$$y_n = x_n + A y_{n-1} + B y_{n-2}$$

may be implemented by setting aside two storage locations: Y1 for y_{n-1}, and Y2 for y_{n-2}, and performing the following sequence;

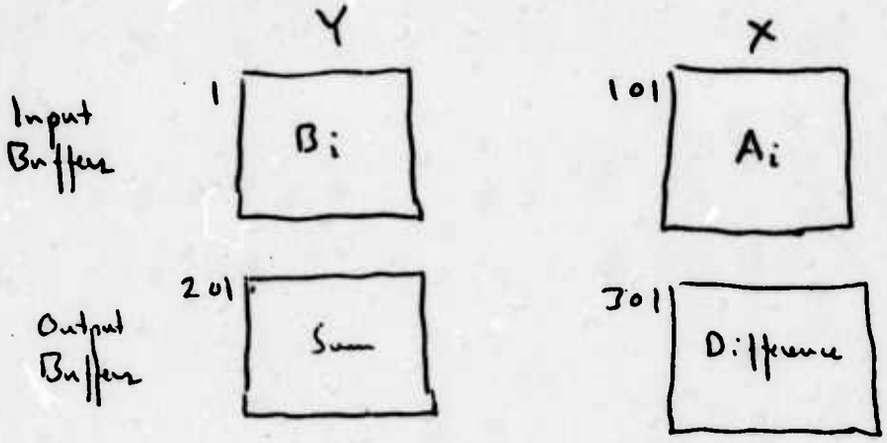
$$x_n + B y_{n-2} \rightarrow T$$
$$y_{n-1} \rightarrow y_{n-2}$$
$$T + A y_{n-1} \rightarrow y_{n-1}$$

which is coded as,

```
MOVE    ,X,10     / put X in Z register
MUL     B,Y2,10   / (X + B·Y2 → Z)
MOVE    ,Y1,Y2    / save Y1 as Y2
MUL     A,Y1,Y1   / (Z + A·Y1 → Y1)
```

Clearly, a coefficient C other than unity for x_n can be handled simply by replacing the first MOVE by a MUL (MUL C, X, 10). It should be noted that at the end of the sequence, the Z register (the addition entry port to the multiplier) has been cleared since the last multiply didn't store in it.

c. Sum and Difference of Two Buffers: For convenience, we assume the four buffers start at locations 1, 101, 201, and 301, and that the length N of the buffers is $\leq$ 100. Also, A_i and B_i can be in "overlapping" X and Y locations, and the Sum and Difference could be both in X or both in Y, or both in both X and Y.



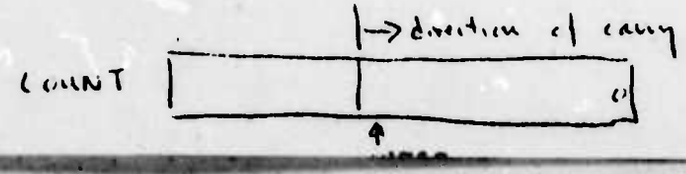

```
                INDEXY    (N, (0 , 12, DOWN     / put N in index registers X and Y, and jump to DOWN
UP      ADD     *  , *100, *200                  /Ai+Bi → Sumi
        SUB     *  , *100, *300                  /Ai-Bi → Diffi
        INDEXY  =  12, (-1, 12                   /decrement index register X by 1 & put the result in index registers X and Y. Skip on C(index register X) = 0
DOWN    MOVE       , 12, 13, UP
                   (One)
```

Note: DOWN must be an even location in the Instruction Memory.

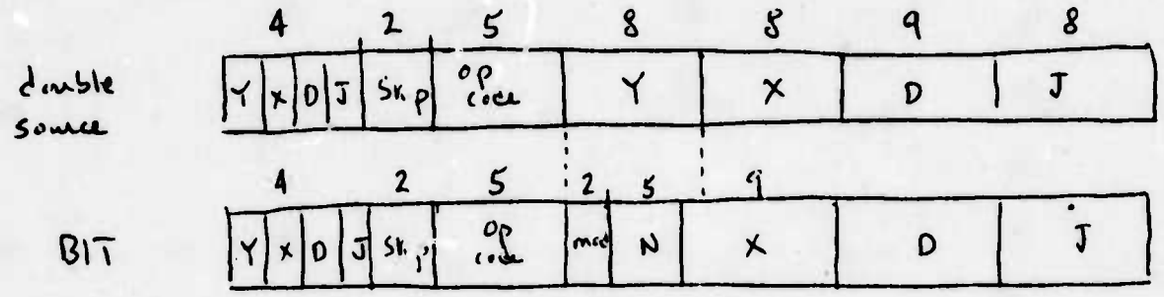
d. Stack Manipulation: In this example, PUSH JUMP and POP RETURN are realized. The subroutine S is assumed to have an associated stack SSTACK and stack pointer SPOINT.

```
Setup           MOVE    , (SSTACK -1, SPOINT    / initialize stack pointer
                :
                :
PUSH JUMP       INDEX D     SPOINT, (1, SPOINT  / increment SPOINT, and put result in register D
                MOVE        , (RETURN , *, S    / store subroutine return address in SSTACK, + start the subroutine S
S               .
                .
                .
                .
POP RETURN      INDEX X     SPOINT, (Ø, TRASH   / put SPOINT in index register X
                INDEX J     (Ø , * , TRASH      / put SPOINT in index register J
                ADD         SPOINT , (-1, SPOINT , *
```

e. Bit-reversed Counter: Given a register, COUNT, containing a number to be incremented in bit reversed order, and another register, WHERE, containing the number of the bit position (LSB=0) at which the count should start:



The following program works because in single-source instructions, if the Y index control bit is 1, the Y index register will be ORed with the bits of the instruction which are in the same position as the 8-bit Y address of double-source instructions:




```
                MOVE    1   , WHERE, 11     / left-shift WHERE by 1,
                                              + put in Y index register
                                              This lines up LSB of
                                              ((WHERE) with LSB of N,
                                              the BIT select field
BACK            BITC    *   , COUNT, COUNT, LOOP  / complement indicated bit,
                                                    + skip if the bit was
                                                    originally a 1
(even)  LOOP                                  / next instruction
                ADD     #11 , (-2 , 11 , BACK  / step right one
                                                 bit, + test the
                                                 next bit
```

BITC (test bit and complement it) will skip until the counting operation is complete. Thus LOOP is the even location reached on counting done, and LOOP+1 contains the instruction that steps the

test bit location to the right, and jumps to the BITC test of the new selected bit.

f. Division: The following mnemonics are assumed:

MOVLO shift in Ones from the right, and shift out through the link

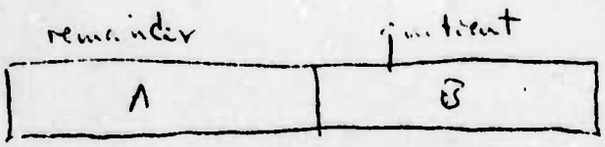
MOVLZ shift in Zeroes from the right, and shift out through the link

MOVL rotate left through the link

(By addition of one instruction to the routine (preset the link), the requirements can be reduced to only the availability of MOVL.)

The following is a routine for the positive integer divide of the double precision number (A,B) by C.

```
        DIV     NEG     , (22   , N , DIVL  / 2's complement of 22₈ into N.
        DIVE    INCRZ   , N, N, DIVL        / add 1 to N and skip on 0
even    LOOP    MOVLO   1, B, B, DOVE       / shift 1 place left
                MOVLZ   1, B, B
                MOVL    1, A, A, DIVE
        DOVE    MOVL    1, T, A , DIVE
even    DIVL    SUBM    (, A, T, LOOP       / Subtract C from A, store the result in T (temporary), and skip on Minus.
                done
```



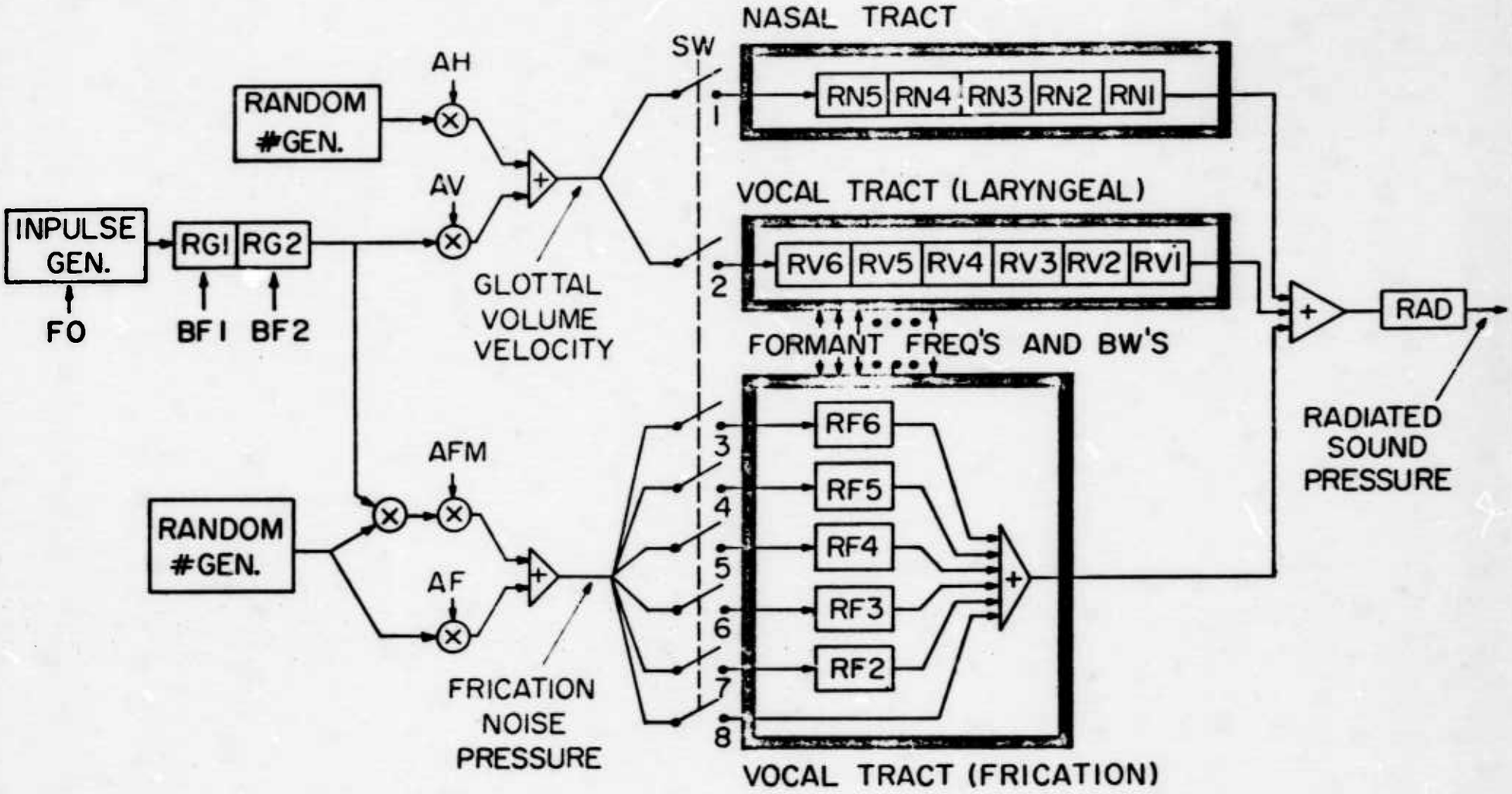

NASAL TRACT
SW
AH
RANDOM #GEN.
AV
INPULSE GEN.
RG1 RG2
FO
BF1 BF2
GLOTTAL VOLUME VELOCITY
RN5 RN4 RN3 RN2 RN1
VOCAL TRACT (LARYNGEAL)
RV6 RV5 RV4 RV3 RV2 RV1
FORMANT FREQ'S AND BW'S
RAD
RADIATED SOUND PRESSURE
AFM
RANDOM #GEN.
AF
FRICATION NOISE PRESSURE
RF6
RF5
RF4
RF3
RF2
VOCAL TRACT (FRICATION)
1
2
3
4
5
6
7
8

Figure 1.

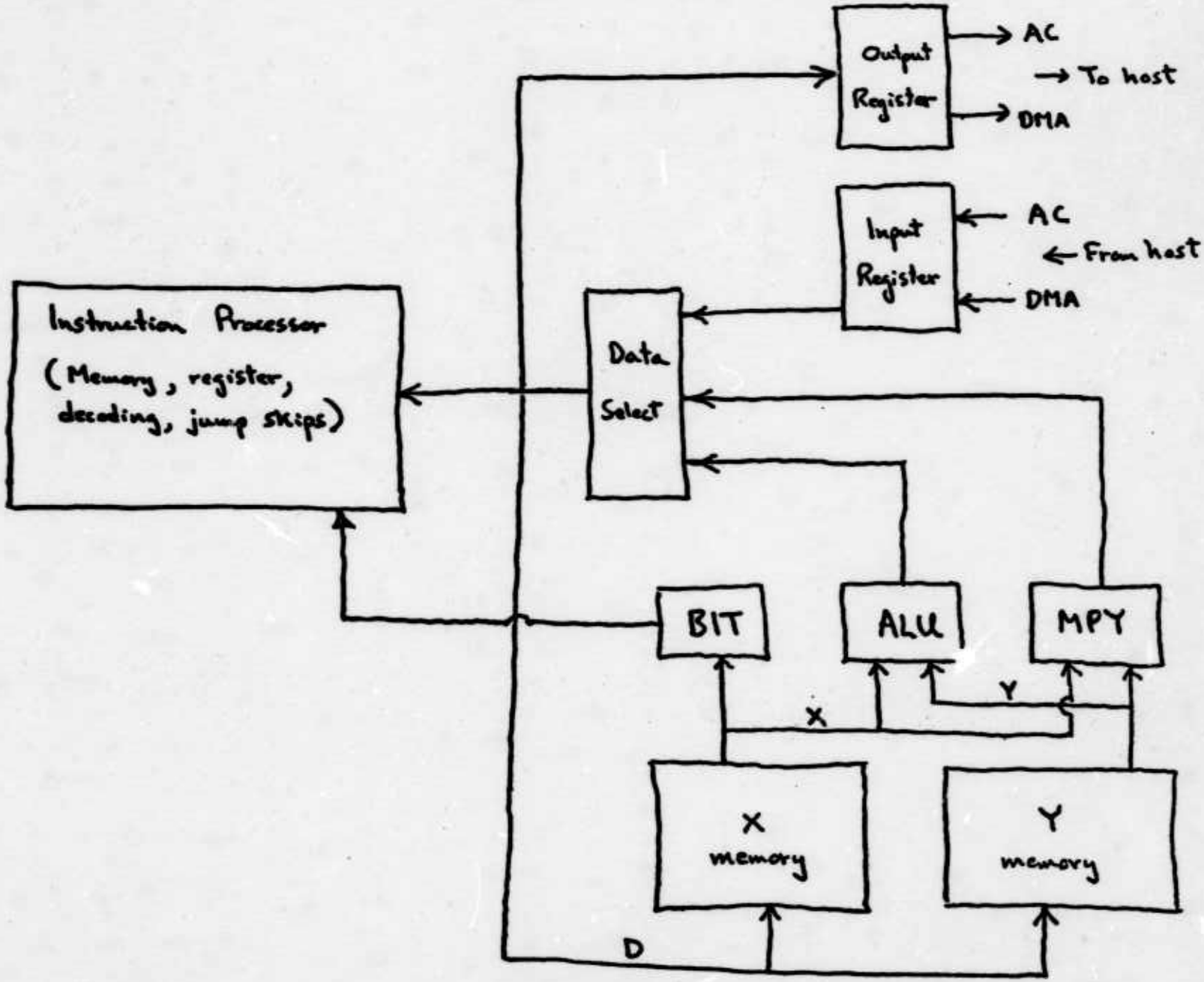

Output Register
AC
To host
DMA
Input Register
AC
From host
DMA
Instruction Processor
(Memory, register, decoding, jump skips)
Data Select
BIT
ALU
MPY
X
Y
X memory
Y memory
D

FIGURE 2

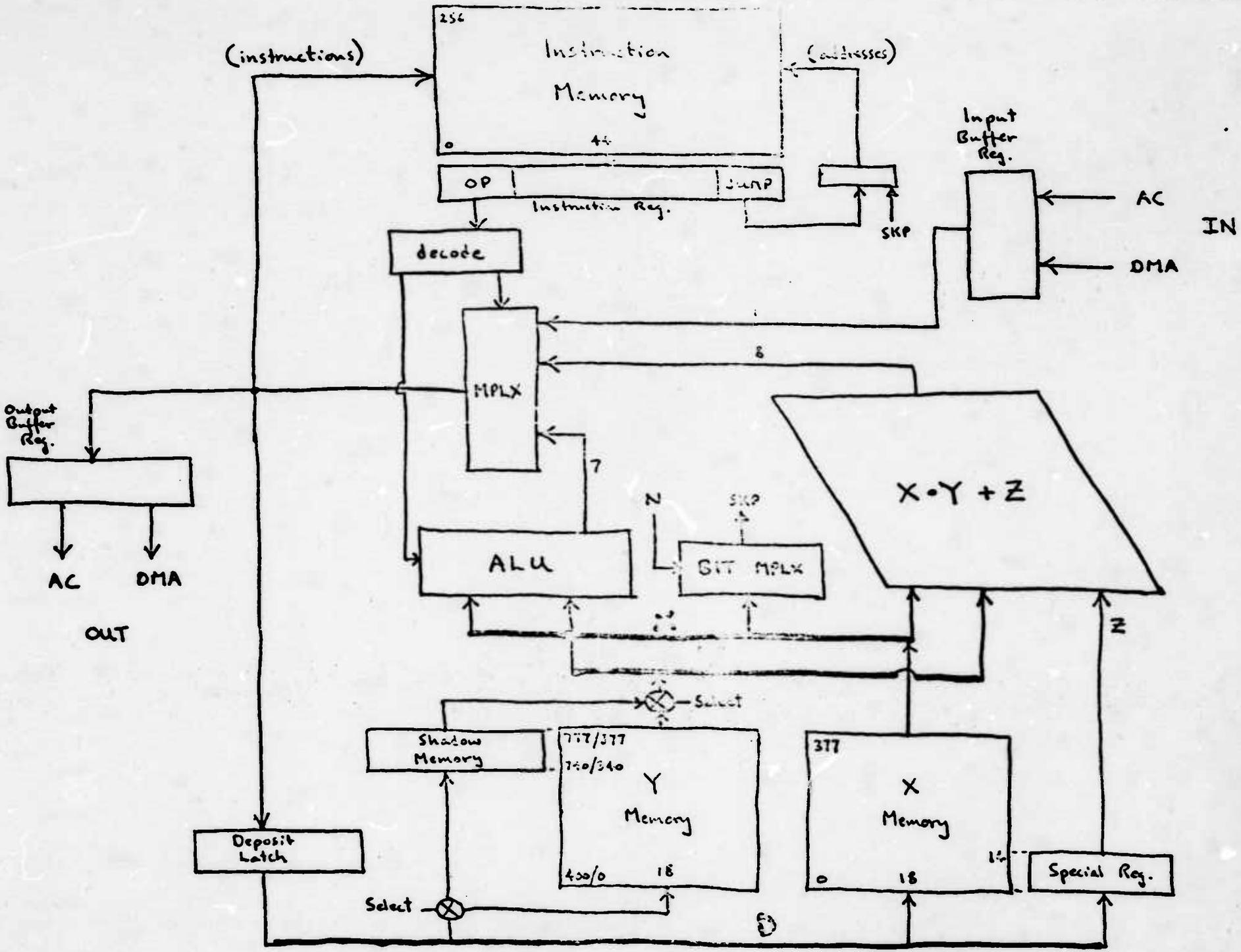
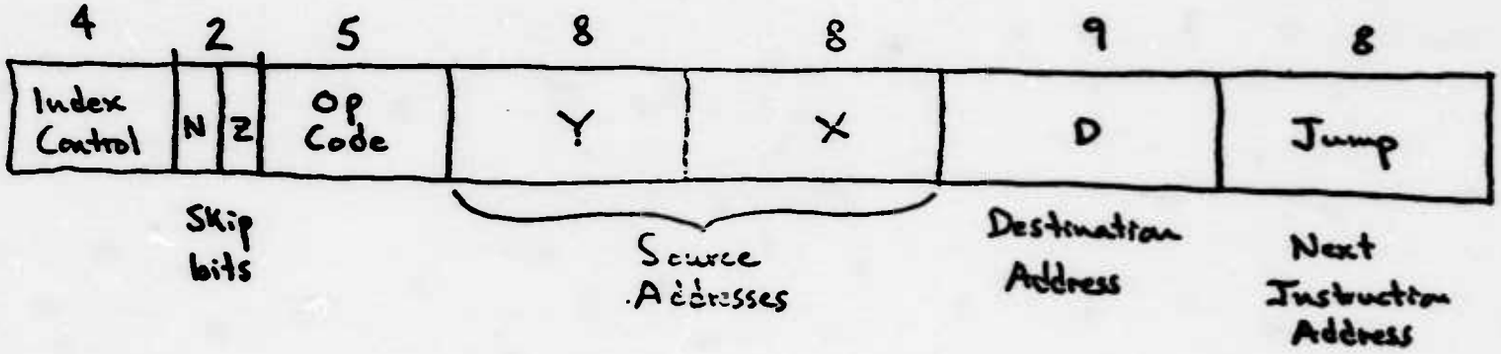

256
Instruction Memory
0
4k
(instructions)
(addresses)
OP
JUMP
Instruction Reg.
SKP
Input Buffer Reg.
AC
DMA
IN
decode
MPLX
8
7
Output Buffer Reg.
AC
DMA
OUT
N
SKP
ALU
SKIP MPLX
X·Y + Z
Z
Select
Shadow Memory
777/377
740/340
Y Memory
400/0
18
377
X Memory
0
18
1k
Special Reg.
Deposit Latch
Select

FIGURE 3

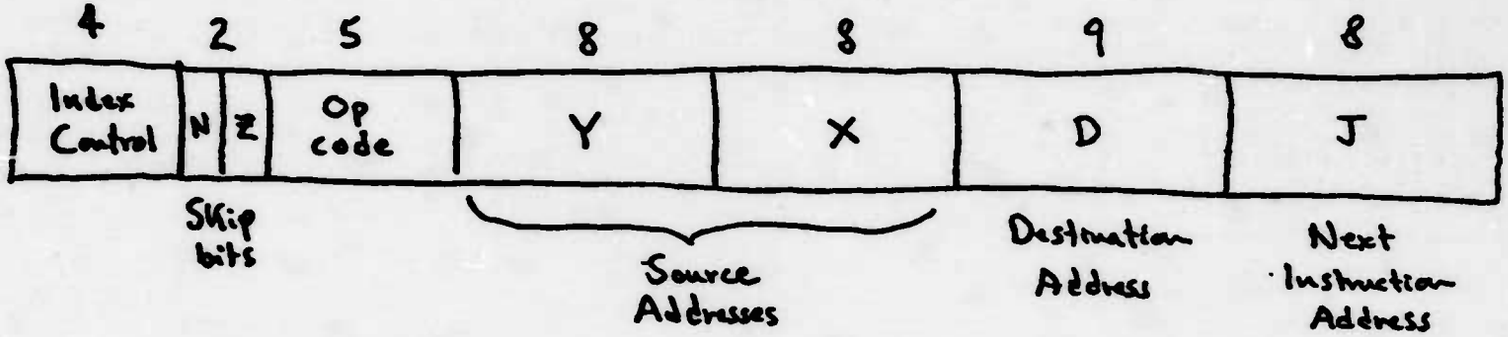
ADD	c(D) ← c(X) + c(Y)
ADDL	c(D) ← c(X) + c(Y) + c(L)
SUB	c(D) ← c(X) − c(Y)
SUBL	c(D) ← c(X) − c(Y) + c(L)
AND	c(D) ← c(X) ∧ c(Y)
IOR	c(D) ← c(X) ∨ c(Y)
XOR	c(D) ← c(X) ⊕ c(Y)



DOUBLE ARGUMENT INSTRUCTION FORMAT

FIGURE 4

$$T \leftarrow c(X) \cdot c(Y) + c(Z)$$
$$Z \leftarrow 0$$
$$c(D) \leftarrow T$$



Scalings Available Through Data Select
(selected by 3 bits of Op code)

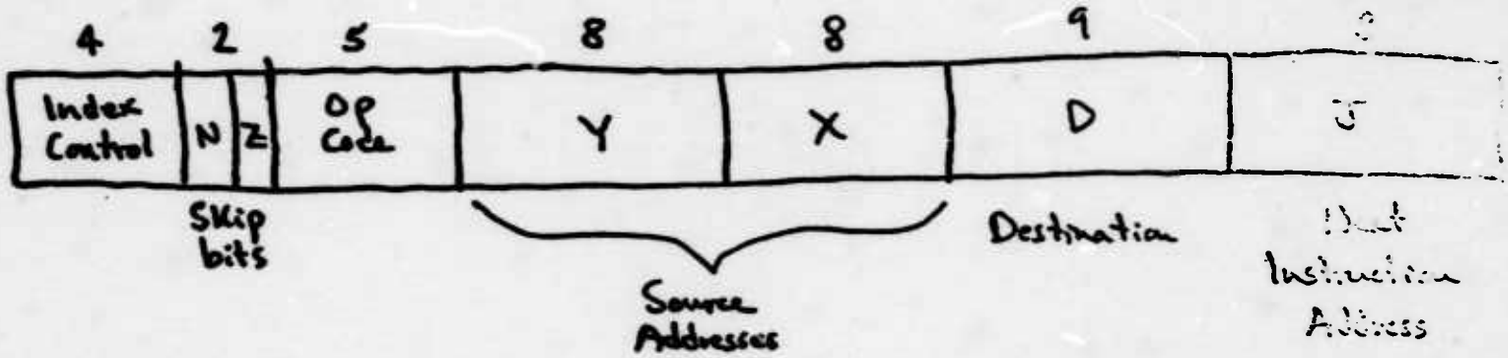
	integer	(LS 18 bits)
	fractional	(MS 18 bits)
1/2	"	
1/4	"	
2	"	
4	"	
8	"	
16	"	

[Possibly, "middle 18 bits", for random number generation.]

MULTIPLY INSTRUCTION FORMAT.

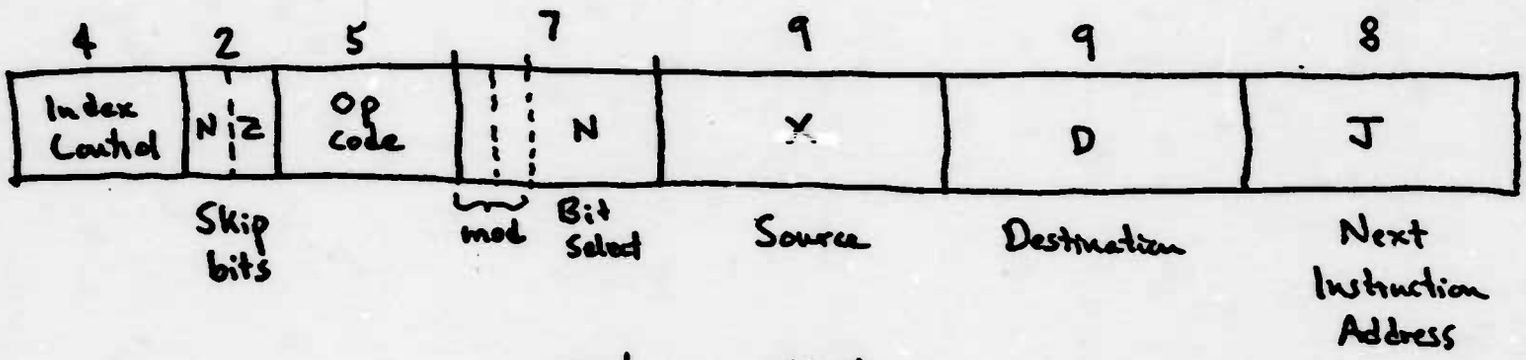
FIGURE 5

INDEX	X	$c(X_X), c(D) \leftarrow c(X) + c(Y)$
INDEX	Y	$c(X_Y), c(D) \leftarrow c(X) + c(Y)$
INDEX	D	$c(X_D), c(D) \leftarrow c(X) + c(Y)$
INDEX	J	$c(X_J), c(D) \leftarrow c(X) + c(Y)$



INDEX INSTRUCTION FORMAT

FIGURE 6

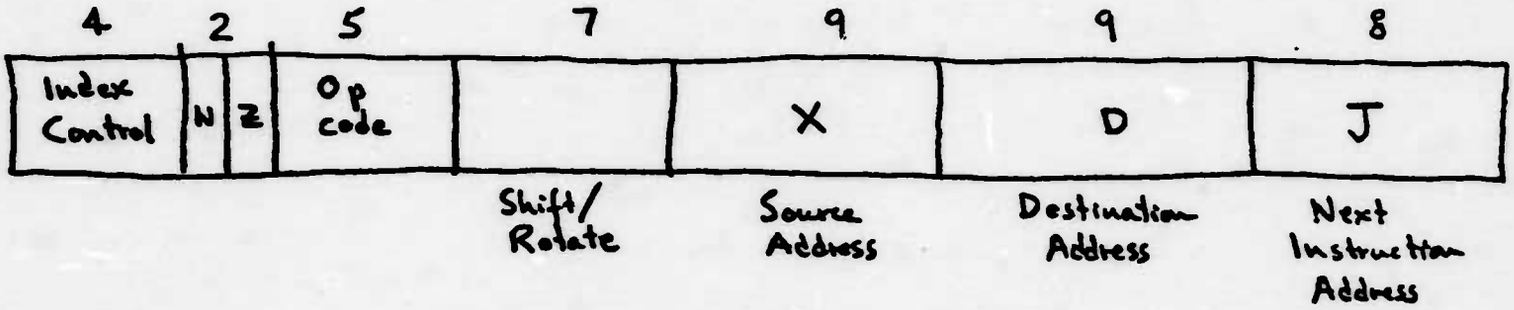


mod	effect
0 0	no change
0 1	clear bit
1 0	set bit
1 1	complement bit

BIT TEST INSTRUCTION FORMAT

FIGURE 7

MOVE	$c(0) \leftarrow c(s)$
INCR	$c(0) \leftarrow c(s) + 1$
DECR	$c(0) \leftarrow c(s) - 1$
COMP	$c(0) \leftarrow \overline{c(s)}$
NEG	$c(0) \leftarrow \overline{c(s)} + 1$



Available Shifts and Rotates

shifts of 0, ±1, ±2, and ±3 bits, modified:

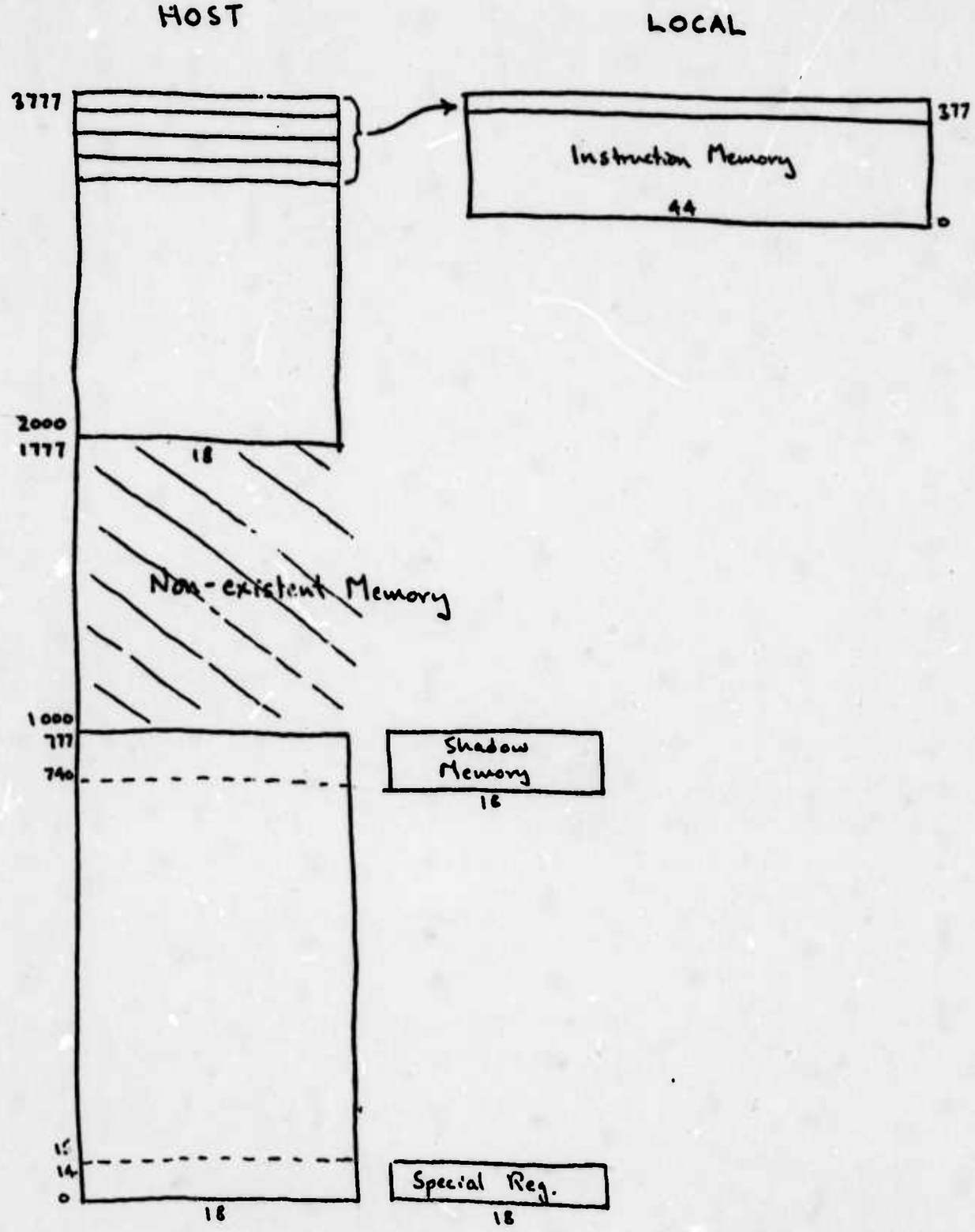
- signed shift
- 1's shift
- 0's shift
- rotate without link
- rotate with link

} both directions

- overflow ⊕ sign (repeated) — right shift only

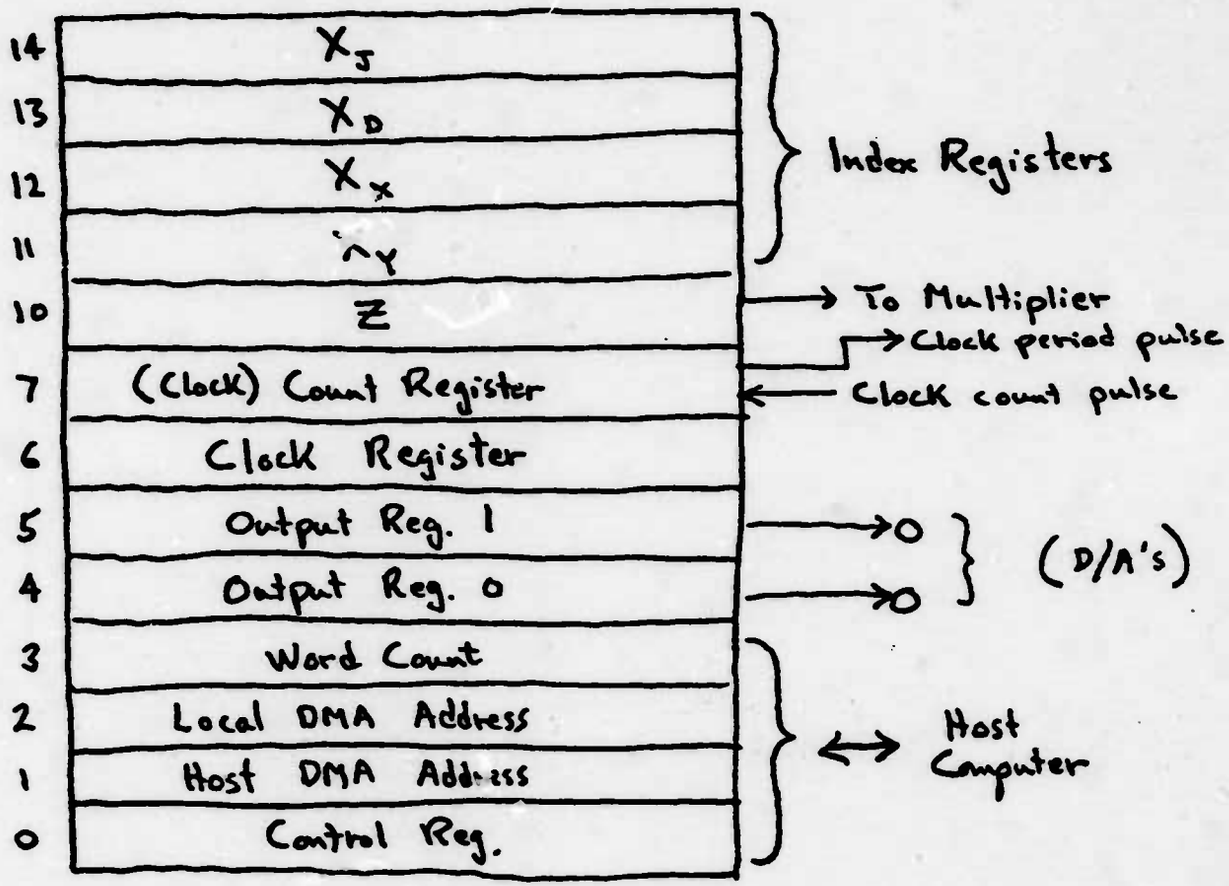
SINGLE SOURCE INSTRUCTION FORMAT

FIGURE 8



ADDRESSING

FIGURE 9



SPECIAL REGISTERS

FIGURE 10

Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform

ALAN V. OPPENHEIM, SENIOR MEMBER, IEEE, AND
CLIFFORD J. WEINSTEIN, MEMBER, IEEE

Invited Paper

***Abstract*—When digital signal processing operations are implemented on a computer or with special-purpose hardware, errors and constraints due to finite word length are unavoidable. The main categories of finite register length effects are errors due to A/D conversion, errors due to roundoffs in the arithmetic, constraints on signal levels imposed by the need to prevent overflow, and quantization of system coefficients. The effects of finite register length on implementations of linear recursive difference equation digital filters, and the fast Fourier transform (FFT), are discussed in some detail. For these algorithms, the differing quantization effects of fixed point, floating point, and block floating point arithmetic are examined and compared.**

The paper is intended primarily as a tutorial review of a subject which has received considerable attention over the past few years. The groundwork is set through a discussion of the relationship between the binary representation of numbers and truncation or rounding, and a formulation of a statistical model for arithmetic roundoff. The analyses presented here are intended to illustrate techniques of working with particular models. Results of previous work are discussed and summarized when appropriate. Some examples are presented to indicate how the results developed for simple digital filters and the FFT can be applied to the analysis of more complicated systems which use these algorithms as building blocks.

I. Introduction

IN PRACTICE, digital signal processing requires the representation of sequence values in a binary format with a finite register length. The effect of the finite word-length constraint manifests itself in several different ways. If a sequence to be processed is derived by sampling an analog waveform, then the finite word-length constraint requires that the analog-to-digital conversion produce only a finite number of values. This represents quantization of the input waveform. Even when we start with data representable with a finite word length, the result of processing will naturally lead to values requiring additional bits for their representation. For example, a b-bit data sample multiplied by a b-bit coefficient results in a product which is $2b$ bits long. If in a recursive digital filter we do not quantize the result of arithmetic operations, the number of bits required will increase indefinitely, since after the first iteration $2b$ bits are required, after the second iteration $3b$ bits are required, etc. The effect of quantization in such a context depends on such factors as whether we are considering fixed-point or floating-point arithmetic, and whether for fixed-point arithmetic we are using a representation of numbers in terms of fractions or integers, or perhaps a mixture. We will be treating the case of fixed-point arithmetic and floating-point arithmetic separately. For fixed-point arithmetic, it is natural in a signal processing context to consider a register as representing a fixed-point fraction. In this way the product of two numbers remains a fraction and the limited register length can be maintained by truncating or rounding the least significant bits. With this type of representation the result of addition on fixed-point fractions need not be truncated or rounded but it can increase in magnitude so that the sum eventually is not a fraction. This effect is commonly referred to as overflow, and can be handled by requiring that the input data be sufficiently small so that the possibility of overflow is avoided. In considering floating-point arithmetic, dynamic range considerations generally can be neglected due to the large range of representable numbers, but quantization is introduced both for multiplication and for addition.

A third effect of finite word length is inaccuracies in parameter values. While generally signal processing parameters are initially specified with unlimited accuracy, they can only be utilized with finite word length. This effect is similar to the effect which arises in implementing analog processing using inaccurate circuit elements. There are two possible approaches to handling the inaccuracies in parameter values. One possibility is to develop design procedures which inherently are insensitive to parameter inaccuracies. An alternate is to choose specifications which are consistent with the limited register length. There is a certain amount that is understood about the effect of inaccuracies in parameter values, but for the most part present results lead to guidelines rather than hard design or analytical strategies.

In the following discussion the relationship between the binary representation of numbers and truncation or rounding is discussed and a statistical model for arithmetic roundoff is presented. This statistical model is then applied to the analysis of fixed-point and floating-point rounding errors in digital filters. The analysis includes a consideration of the effect of dynamic range in developing and comparing signal-to-noise ratios for fixed-point and floating-point filters. It is not always possible to treat the effects of arithmetic roundoff in terms of a simple statistical model. Some approaches and results are available in the literature on the limit cycle behavior of digital filters due to arithmetic roundoff, and a discussion of some of these results is included.

For the analysis of arithmetic roundoff in computation of the discrete Fourier transform using the fast Fourier trans-

Manuscript received May 11, 1972. A. V. Oppenheim was supported in part by the National Science Foundation under Grant GK-31353 and in part by the Advanced Research Project Agency of the Department of Defense, monitored by ONR under Contract N00014-67-A-0204-0064; C. J. Weinstein was supported in part by the U. S. Air Force. *This invited paper is one of a series planned on topics of general interest—The Editor.*

A. V. Oppenheim is with the Department of Electrical Engineering and the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

C. J. Weinstein is with Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass. 02173.

form (FFT) algorithm a statistical model is used. With this model the signal-to-noise ratio is developed and compared for fixed-point and floating-point arithmetic.

While for any given filter configuration or spectral analysis problem it can be difficult to carry out a detailed analysis of the effects of finite register length there are a number of general guidelines that can be distilled from the results presented here. In Section IV some examples and guidelines are presented for filters implemented with fixed-point arithmetic and with floating-point arithmetic as well as for filters implemented with the FFT.

This paper is intended primarily as a tutorial review of a subject which has received considerable attention over the past few years. The analyses which are presented here are selected to illustrate techniques of working with particular models. Previous work is freely referenced, discussed, and borrowed from.

II. Number Representation and Its Effect on Quantization

A. Fixed-Point and Floating-Point Numbers

The manner in which finite word-length effects are manifested is closely tied to the way in which numbers are represented.

Digital computers and special purpose digital hardware for the most part use a number representation with a radix of 2, i.e., a binary representation. Therefore, a number is represented by a sequence of binary digits which are either zero or unity. Just as a decimal number is represented as a string of decimal digits with a decimal point dividing the integer part from the fractional part, the sequence of binary digits is divided by a binary point into those representing the integer part of the number and those representing the fractional part. Thus if Δ denotes the location of the binary point, the binary number $1001_\Delta 0110$ has the decimal value of $(1\times 2^3+0\times 2^2+0\times 2^1+1\times 2^0)+(0\times 2^{-1}+1\times 2^{-2}+1\times 2^{-3}+0\times 2^{-4})$. This representation always corresponds to a positive number.

The manner in which arithmetic is implemented in a digital computer or in a special purpose hardware depends on where in the register the binary point is located. For fixed-point arithmetic, the implementation is based on the assumption that the location of the binary point is fixed. The manner in which addition is carried out will not depend on the location of the binary point for fixed-point arithmetic as long as it is the same for every register. For multiplication, however, the location of the binary point must be known. For example, consider the product of the two 4-bit numbers 1001_Δ and 0011_Δ. In general, of course, the product of two b-bit numbers will be $2b$ bits long. The 8-bit product of the above number is 00011011_Δ. If, on the other hand, we consider the 4-bit fractions $_\Delta 1001$ and $_\Delta 0011$, then the 8-bit product is $_\Delta 00011011$. In digital filtering applications, it is usually necessary to approximate the $2b$-bit product of two b-bit numbers by a b-bit result. In integer arithmetic this is difficult. With fractional arithmetic, on the other hand, this can be accomplished by truncating or rounding to the most significant b bits. For multiplication with fractions, overflow can never occur since the product of two fractions is a fraction. Thus for the 4-bit example previously mentioned, the product $_\Delta 00011011$ can be approximated by $_\Delta 0001$ (truncation) or $_\Delta 0010$ (rounding).

An alternative to fixed-point arithmetic is a floating-point representation. In this case, a positive number F is represented as $F=2^cM$, where M, the mantissa, is a fraction between 1/2 and 1, and c, the characteristic, can be either positive or negative. The product of two floating-point numbers is carried out by multiplying the mantissa as fixed-point fractions and adding the characteristics. Since the product of the mantissas will be between 1/4 and 1, a normalization of the mantissa and corresponding adjustment of the characteristic may be necessary. The sum of two floating-point numbers is carried out by scaling the mantissas of the smaller number to the right until the characteristics of the two numbers are equal and then adding the mantissas. For example, consider the sum of F_1 and F_2 with $F_1=4$ and $F_2=5/4$. Then in floating-point notation, $F_1=2^{c_1}M_1$, and $F_2=2^{c_2}M_2$ with

$$c_1 = 11_\Delta \quad (=3 \text{ decimal})$$
$$M_1 = {}_\Delta 1000 \quad (=0.5 \text{ decimal})$$
$$c_2 = 1_\Delta \quad (=1 \text{ decimal})$$
$$M_2 = {}_\Delta 1010 \quad (=5/8 \text{ decimal}).$$

In order to carry out the addition, c_2 must be changed to equal c_1 and M_2 must be adjusted accordingly. Thus first the representation of F_2 is changed to $F_2=2^{\hat{c}_2}\hat{M}_2$ with

$$\hat{c}_2 = 11_\Delta$$
$$\hat{M}_2 = {}_\Delta 00101$$

in which case the mantissas can now be added. The resulting sum is $F=2^cM$ with $c=11_\Delta$ and $M={}_\Delta 10101$. In this case the sum of M_1 and $\hat{M}_2$ is a fraction between 1/2 and 1 and therefore no further adjustment of c has to be carried out. In a more general case, the sum may not be in that range, and consequently, c would be adjusted to bring the mantissa into the proper range. From this example it should be clear that in general with floating-point arithmetic, the mantissa can exceed the register length and must therefore be truncated or rounded for both addition and multiplication whereas this is only necessary for multiplication in the fixed-point case. On the other hand, if the result of addition in the fixed-point case exceeds the register length, truncation or rounding will not help, i.e., the dynamic range has been exceeded. Thus while floating point introduces error due to arithmetic round-off, it provides much greater dynamic range than fixed point. As we will see later, both of these effects must be considered when comparing fixed-point and floating-point realizations of digital filters.

B. Representation of Negative Numbers

There are three common means used for representing fixed-point negative numbers. The first, and most familiar, is sign and magnitude, i.e., the magnitude (which is of course positive) is represented as a binary number and the sign is represented by the leading binary digit which, if 0 corresponds to a + and if 1 corresponds to a − (or vice versa). Thus for example, in sign and magnitude $0_\Delta 0011$ represents 3/16 and $1_\Delta 0011$ represents −3/16. Two other related representations of negative numbers are often referred to as one's-complement and two's-complement representations. Considering all numbers to be fractions, a positive number is represented as before. For two's complement representation a negative number is represented by 2.0 minus its magnitude. For example $-(0_\Delta 0110)$ in sign and magnitude is represented as $1_\Delta 1010$ in two's-complement since $10_\Delta 000-0_\Delta 0100=1_\Delta 1010$. For one's-complement, the negative number is represented by subtracting the magnitude from the largest number representable in the register. Thus $-(0_\Delta 0110)$ is represented by

$(1_\Delta 1111) - (0_\Delta 0110) = 1_\Delta 1001$. One's complement representation is equivalent to representing a negative number by the bit-by-bit complement of its magnitude. The choice of representation for negative numbers in a particular system is usually based almost entirely on hardware considerations.

For the representation of negative floating-point numbers there are a variety of conventions that have been used. In this paper we will consider the sign of the number to be associated with the mantissa so that the mantissa is a signed fraction. The representation of this signed fraction can of course be in sign and magnitude, one's-complement of two's-complement notation.

C. A Model for Arithmetic Roundoff

In formulating a model for arithmetic roundoff, we shall consider both fixed-point numbers and mantissas of floating-point numbers to be represented as $b+1$-bit binary fractions, with the binary point just to the right of the highest order bit (or sign bit). This convention represents no loss of generality, and its convenience has been alluded to above. The numerical value (for positive numbers) of a one in the least significant bit is 2^{-b}, and this quantity can be referred to as the width of quantization.

As indicated previously, the effect of finite register length on the result of arithmetic operations depends on whether fixed-point or floating-point arithmetic is used, and how negative numbers are represented. Let us consider first the effect of truncation and rounding in the fixed-point case. For sign and magnitude, one's-complement and two's-complement, the representation of positive numbers is identical and, consequently, so is the effect of truncation and rounding. If E_T denotes the error due to truncation, i.e., the value after truncation minus the value before truncation, this error will always be negative for positive numbers. That is, the effect of truncation is to reduce the value of the numbers. More specifically, if b_2 denotes the number of bits (exclusive of sign) after truncation, and b_1 denotes the number of bits before truncation, then the result satisfies $0 \geq E_T \geq -(2^{-b_2} - 2^{-b_1})$.

With sign and magnitude representation of negative numbers, truncation reduces the magnitude of the number and the error E_T satisfies $0 \leq E_T \leq (2^{-b_2} - 2^{-b_1})$. For a two's-complement negative number represented by the bit string $1_\Delta, a_1, a_2, \cdots, a_{b_1}$, the magnitude is given by

$$M_1 = 2.0 - x_1$$

where

$$x_1 = 1 + \sum_{i=1}^{b_1} a_j 2^{-j}.$$

Truncation to b_2 bits ($b_2 < b_1$) produces the bit string $1_\Delta, a_1, a_2, \cdots, a_{b_2}$, where now the magnitude is

$$M_2 = 2.0 - x_2$$

with

$$x_2 = 1 + \sum_{i=1}^{b_2} a_i 2^{-i}.$$

The change in magnitude is

$$\Delta M = M_2 - M_1 = \sum_{i=b_2+1}^{b_1} a_j 2^{-j}$$

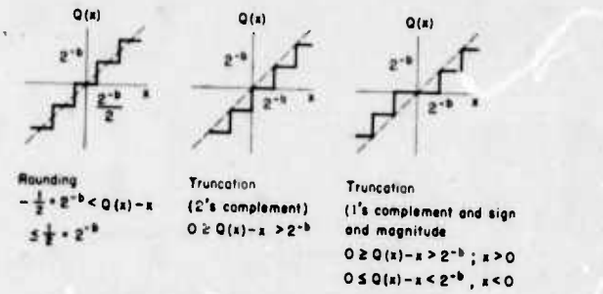


Fig. 1. Transfer characteristics for rounding and truncation.

and it is easily seen that

$$0 \leq \Delta M \leq 2^{-b_2} - 2^{-b_1}.$$

Hence the effect of truncation for two's-complement negative numbers is to *increase* the magnitude of the negative number; the truncation error is negative, and satisfies $0 \geq E_T \geq -(2^{-b_2} - 2^{-b_1})$.

For a one's-complement negative number represented by the bit string $1_\Delta, a_1, a_2, \cdots, a_{b_1}$, the magnitude is given by

$$M_1 = 2.0 - 2^{-b_1} - x_1$$

and truncation to b_2 bits yields a magnitude

$$M_2 = 2.0 - 2^{-b_2} - x_2$$

where x_1 and x_2 are as defined above. The change in magnitude is

$$\Delta M = M_2 - M_1 = \sum_{i=b_2+1}^{b_1} a_i 2^{-i} - (2^{-b_2} - 2^{-b_1})$$

and now

$$0 \geq \Delta M \geq -(2^{-b_2} - 2^{-b_1}).$$

Hence the effect of truncation for one's complement negative numbers is to *decrease* the magnitude of the negative number; the truncation error is positive, and satisfies $0 \leq E_T \leq 2^{-b_2} - 2^{-b_1}$.

The effect of rounding, of course, will be the same independent of how negative numbers are represented and the rounding error will always be greater than or equal to $(-1/2)2^{-b}$ and less than or equal to $(+1/2)2^{-b}$. The effect of truncation and rounding for the fixed-point case is summarized in Fig. 1 where x represents the value before truncation or rounding and $Q(x)$ represents the value after. In the figure it is assumed that x can take on a continuous range of values, corresponding to $b_1 = \infty$ in the discussion above, and that the quantized word length is b bits plus sign.

For the case of floating-point arithmetic, the effect of truncation or rounding is reflected only in the mantissa. It is convenient in the floating-point case to describe the error in a multiplicative sense rather than in an additive sense as is done in fixed-point arithmetic. In other words, for a floating-point word, if x represents the value before truncation or rounding and $Q(x)$ represents the value after, then we express $Q(x)$ as equal to $x(1+\epsilon)$. For the case of rounding, for example, the error in the mantissa is between $\pm 2^{-b}/2$, and consequently the error in the value of the floating-point word is

$$-2^c \cdot \frac{2^{-b}}{2} \leq Q(x) - x \leq 2^c \cdot \frac{2^{-b}}{2}$$

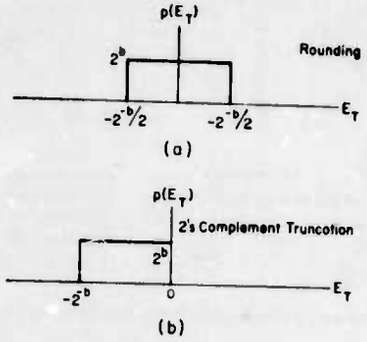


Fig. 2. (a) Probability density function for rounding noise. (b) Probability density function for noise due to two's-complement truncation.

or, since $Q(x)-x=\epsilon x$

$$-2^{c}\cdot\frac{2^{-b}}{2}\leq \epsilon x \leq 2^{c}\cdot\frac{2^{-b}}{2}$$

and since $2^{c-1}\leq x<2^{c}$, we can write that for the case of rounding $-2^{-b}\leq\epsilon\leq 2^{-b}$. In a similar manner we can show that for one's-complement and for sign and magnitude truncation $0\geq\epsilon\geq -2\cdot 2^{-b}$. For two's-complement truncation

$$0\geq \epsilon \geq -2\cdot 2^{-b}, \qquad x>0$$
$$0\leq \epsilon \leq 2\cdot 2^{-b}, \qquad x<0.$$

D. Statistical Model of Arithmetic Roundoff

A convenient means for analyzing the effect of quantization is to represent the error statistically [1], [2]. In particular, for the case of fixed-point arithmetic and rounding E_T is represented as a random variable with a probability density shown in Fig. 2(a). For the case of two's-complement truncation, the probability density is shown in Fig. 2(b).

In each of these cases, the assumption is that the random variable E_T is independent of x. For one's complement and sign magnitude truncation, this assumption cannot be made since the mean value of the error is directly correlated with the sign of x. In the analysis that follows for fixed-point arithmetic, the discussion is phrased in terms of rounding. The results are easily modified for two's-complement truncation. In particular the variance of the noise is identical for both cases. However, for rounding the noise is zero mean and for two's-complement truncation it is not zero mean.

For the floating-point case, the parameter ϵ is considered to be a random variable which is independent of x. In that case the assumption of independence is reasonable for rounding, sign and magnitude truncation, and one's-complement truncation, but not for two's-complement truncation. The random variable ϵ is bounded by $-2^{-b}\leq\epsilon\leq 2^{-b}$. We will generally assume ϵ to be uniformly distributed in this range with a variance $\sigma_\epsilon^2=(1/3)2^{-2b}$. Empirical work has shown that the distribution is not quite uniform so that while σ_ϵ^2 is proportional to 2^{-2b}, the constant of proportionality is slightly less than 1/3. However, the interpretation of the results depends primarily on the proportionality to 2^{-2b}.

III. Finite Register Length Effects for Digital Filters [3]

A. Introduction

The basic arithmetic operations involved in implementation of a digital filter are multiplication by a constant and addition. For fixed-point arithmetic, roundoff is introduced only after the multiplication. Because of the possibility of overflow due to addition, there is a dynamic range limitation in fixed-point filters. In contrast, floating-point filter implementation has a much less severe dynamic range constraint, although arithmetic roundoff is introduced due to both multiplication and addition. In the next sections we will first develop the statistical analysis of arithmetic roundoff for fixed-point filters including dynamic range considerations. This is followed by a statistical analysis for floating-point arithmetic and a discussion of zero input limit cycle behavior for fixed-point arithmetic.

B. Statistical Analysis of Fixed-Point Errors in a Digital Filter [4], [5]

In many situations it is reasonable to model the effect of rounding in a digital filter by a simple statistical model. The approach is to model the effect of the rounding at each multiplier by a white-noise source uniformly distributed in amplitude between plus and minus $(1/2)2^{-b}$. Each of the noise sources is assumed to be linearly independent of each other and of the input. Experimentally these assumptions have been justified for a broad class of inputs including random signals, speech, etc. The model is clearly not valid for certain inputs, such as constant inputs. If the impulse response from the kth noise source to the output is $h_k(n)$ then the steady-state output noise variance due to the kth noise source is

$$\sigma_{ok}^2=\sigma_\epsilon^2\sum_{n=0}^{\infty}h_k^2(n) \tag{1}$$

where $\sigma_\epsilon^2=(1/12)2^{-2b}$. Since all the noise sources are assumed to be uncorrelated, the total output noise is

$$\sigma_o^2=\sum_k\sigma_{ok}^2. \tag{2}$$

For example, if we consider the first-order filter in Fig. 3 one noise source is introduced. In this case, the impulse response from the noise source input to the output is $h(n)=a^nu_{-1}(n)$ where u_{-1} denotes a unit step sequence, so that

$$\sigma_o^2=\frac{1}{12}2^{-2b}\frac{1}{1-a^2}. \tag{3}$$

For a second-order filter with one complex pole pair there are two noise sources as indicated in Fig. 4. The resulting output noise is

$$\sigma_o^2=\frac{2}{12}2^{-2b}\left(\frac{1+r^2}{1-r^2}\frac{1}{r^4+1-2r^2\cos 2\theta}\right). \tag{4}$$

C. Dynamic Range Considerations for Fixed-Point Filters

As indicated previously, the possibility of overflow must be considered in the implementation of digital filters with fixed-point arithmetic. With the convention that each fixed-point register represents a signed fraction, each node in the filter must be constrained to maintain a magnitude less than unity in order to avoid overflow. Letting $x(n)$ denote the filter input and $y_k(n)$ and $h_k(n)$ denote the output and unit sample response for the kth node in the filter, then

$$y_k(n)=\sum_{r=0}^{\infty}h_k(r)x(n-r). \tag{5}$$

If $x_{\max}$ denotes the maximum of the absolute value of the in-

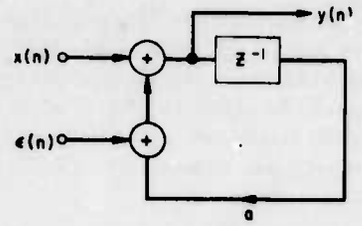


Fig. 3. Noisy first-order filter (fixed point).

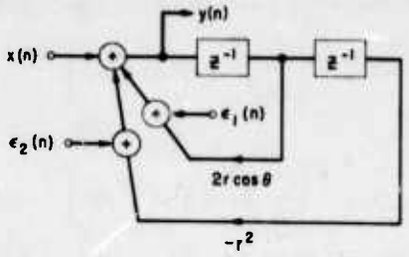


Fig. 4. Noisy second-order filter (fixed point).

put then

$$|y_k(n)| \leq x_{\max} \sum_{r=0}^{\infty} |h_k(r)|. \tag{6}$$

Thus, since we require that $|y_k(n)| < 1$, (6) requires that

$$x_{\max} < 1 \Big/ \sum_{r=0}^{\infty} |h_k(r)|, \qquad \text{for all } k. \tag{7}$$

Equation (7) thus provides an upper bound on the maximum value of the input to insure that no overflow occurs in the kth node. For a general input (7) in fact provides a least upper bound, i.e., if the maximum value of the input exceeds the bound, overflow can occur. This is a consequence of the fact that equality can be achieved in (6) with a sequence $x(n)$ for which at $n = n_o$, $x(n_o - r) = [\operatorname{sgn} h_k(r)]$ for $r = 0$ to ∞. (Where $\operatorname{sgn}(x) = 1$ for $x \geq 0$ and $\operatorname{sgn}(x) = -1$ for $x < 0$.) Thus in the most general case, (7) is required to guarantee that no overflow occurs. The condition in (7) would generally be satisfied by applying attenuation to the signal at the filter input.

If we assume, for example, that the input $x(n)$ is a white-noise sequence with a uniform amplitude distribution, we would choose for the case of the first-order filter a maximal input amplitude of $(1-a)$. For this case, if σ_x^2 denotes the variance of the input signal, and σ_y^2 denotes the variance of the output signal, then

$$\sigma_x^2 = \left(\frac{1}{3}\right)(1-a)^2 \tag{8a}$$

$$\sigma_y^2 = \left(\frac{1}{3}\right)\left(\frac{(1-a)^2}{1-a^2}\right). \tag{8b}$$

For this example, we can then compute a *noise-to-signal* ratio as the ratio σ_o^2/σ_y^2 with the result

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{1}{4}\, 2^{-2b}\, \frac{1}{(1-a)^2}. \tag{9}$$

In a similar manner we can derive a noise-to-signal ratio for the second-order filter shown in Fig. 4. As in the first-order case, we restrict the maximum input in order to guarantee that the dynamic range of the registers is not exceeded. If we consider the input sequence to be uniformly distributed white noise, the resulting output noise-to-signal ratio will be

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{1}{2}\, 2^{-2b}\left(\sum_{n=0}^{\infty} |h_n|\right)^2 = \frac{1}{2}\, 2^{-2b}\left(\frac{1}{\sin\theta} \sum_{n=0}^{\infty} r^n |\sin[(n+1)\theta]|\right)^2. \tag{10}$$

While it is difficult to evaluate this expression exactly, it is possible to obtain an upper and lower bound. Since $\sum_{n=0}^{\infty} |h_n|$ is the largest possible output obtainable with an input that never exceeds unity, it must be larger than the response of the second-order filter to a sinusoid of unity amplitude at the resonant frequency. With this consideration, we can write that

$$\left(\sum_{n=0}^{\infty} |h_n|\right)^2 \geq 1/(1-r)^2(1 + r^2 - 2r\cos 2\theta) \tag{11}$$

since the right-hand side of this inequality is the gain at resonance. Furthermore,

$$\left(\frac{1}{\sin\theta} \sum_{n=0}^{\infty} r^n |\sin[(n+1)\theta]|\right)^2 \leq \left(\frac{1}{\sin\theta} \sum_{n=0}^{\infty} r^n\right)^2. \tag{12}$$

Therefore, for the second-order case

$$\frac{1}{2}\, 2^{-2b}\, \frac{1}{(1-r)^2(1 + r^2 - 2r\cos 2\theta)} \leq \frac{\sigma_o^2}{\sigma_y^2} \leq \frac{1}{2}\, 2^{-2b}\, \frac{1}{\sin^2\theta(1-r)^2}. \tag{13}$$

For both the first- and second-order filter an expression for the noise-to-signal ratio can be obtained which provides some insight into the behavior of the noise-to-signal ratio as the poles approach the unit circle. For the first-order filter let $\delta = 1 - a$ so that as $\delta \to 0$, the pole approaches the unit circle. Then in terms of δ, the noise-to-signal ratio for the first-order filter is

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{1}{4}\, 2^{-2b}\, \frac{1}{\delta^2}. \tag{14}$$

For the second-order filter, let $\delta = 1 - r$ so that, again, as $\delta \to 0$ the poles approach the unit circle. Then if we assume that $\delta \ll 1$, we can approximate $(1 + r^2 - 2r\cos 2\theta)$ as

$$(1 + r^2 - 2r\cos 2\theta) \cong 4\sin^2\theta + \delta^2 \tag{15}$$

which for $4\sin^2\theta$ large compared with δ^2 we will approximate as $4\sin^2\theta$. Consequently, incorporating this approximation,

$$\left(\frac{1}{2}\right) 2^{-2b}\, \frac{1}{4\delta^2\sin^2\theta} \leq \frac{\sigma_o^2}{\sigma_y^2} \leq \frac{1}{2}\, 2^{-2b}\, \frac{1}{\delta^2\sin^2\theta}. \tag{16}$$

Thus we observe that the noise-to-signal ratio as considered thus far can be considered to be proportional to $2^{-2b}/\delta^2$. We note from this dependence that if δ is halved, then to maintain the same noise-to-signal ratio b must be increased by 1, i.e., one bit must be added to the register length. This dependence provides a convenient basis for comparison of different overflow strategies and different kinds of arithmetic.

In the above analysis, the filter input was assumed to be uniformly distributed white noise. As δ approaches zero the frequency response of both the first- and second-order filter

becomes more selective so that more and more of the input energy is out of band. An alternative basis for determining the noise-to-signal ratio is for an input which is sinusoidal. For this choice of inputs, of course, we would not use the general condition of (7) to avoid overflow since we can determine exactly the maximum allowable input amplitude as a function of the filter parameters.

In particular, if the input is of the form $x(n)=x_{\max}\cos n\phi$ then the steady-state output is of the form $y(n)=y_{\max}\cos(n\phi+\psi)$. To prevent overflow, $y_{\max}$ must be less than unity and to maximize the output signal energy, $y_{\max}$ is chosen to be as large as possible. Thus the maximum noise-to-signal ratio is obtained when $x_{\max}$ is chosen so that $y(n)=\cos(n\phi+\psi)$. Note that in order to choose $x_{\max}$ in this way, the frequency of the input signal must be known. For an input sinusoid of unknown frequency $x_{\max}$ must be attenuated so that overflow will not occur even in the worst case, where the frequency of the input coincides with the peak gain in the filter's transfer function.

For fixed-point filters, within the validity of the statistical model for roundoff error, the output noise is independent of the form and amplitude of the input signal. Thus for this choice of inputs, the noise-to-signal ratio obtained for the first-order filter is

$$\frac{\sigma_o^2}{\sigma_y^2}=\frac{1}{24}2^{-2b}\frac{1}{1-a^2}. \tag{17}$$

If, as before, we let $a=1-\delta$, then for $\delta\ll 1$

$$\frac{\sigma_o^2}{\sigma_y^2}=\frac{1}{48}\frac{2^{-2b}}{\delta}. \tag{18}$$

Thus in this case, the noise-to-signal ratio is proportional to $1/\delta$ rather than $1/\delta^2$ so that if δ is multiplied by $1/4$ and the register length is increased by one bit, the noise-to-signal ratio will remain constant. We can consider the second-order case in a similar manner. Again for a sinusoidal input, the output with maximum amplitude has the form $y(n)=\cos(n\phi+\psi)$ so that the noise-to-signal ratio in this case is

$$\frac{\sigma_o^2}{\sigma_y^2}=\frac{1}{12}2^{-2b}\left(\frac{1+r^2}{1-r^2}\,\frac{1}{1+r^4-2r^2\cos 2\theta}\right). \tag{19}$$

Again, choosing $r=1-\delta$ with $\delta\ll 1$,

$$\frac{\sigma_o^2}{\sigma_y^2}\cong\frac{2^{-2b}}{4\delta\sin^2\theta} \tag{20}$$

so that, as with the first-order filter, the noise-to-signal ratio is proportional to $1/\delta$ rather than $1/\delta^2$. The comparison in the noise-to-signal ratio for a white-noise input and a sinusoidal input serves to illustrate the dependence of the effect of dynamic range considerations on the particular form of the input. In some sense, the two cases considered represent extremes. As the input becomes more confined to a known narrow band of frequencies the above analysis with a sinusoidal input would be more representative, and as the input becomes more wide-band the above analysis with a white-noise input is more representative.

In the above discussion, the noise-to-signal ratio for the case of white-noise input was derived on the basis that overflow must be avoided. In a practical case, a scaling of the input on the basis of (7) can be considered to be somewhat pessimistic since the probability of equality being attained in (7) is extremely small. Furthermore, for many filters it is difficult to compute the sum in (7). Jackson [7] has formulated the dynamic range constraints on fixed-point digital filters in terms of L_p norms. In particular, let $Y(\omega)$, $X(\omega)$, and $H(\omega)$ denote the Fourier transforms of the filter output, input, and system impulse response, respectively. Then it can be shown in general that

$$|y(n)|\le\|H\|_p\|X\|_q \quad 1/p+1/q=1 \tag{21}$$

where $\|H\|_p$ and $\|X\|_q$ are the L_p norm and L_q norm of $H(\omega)$ and $X(\omega)$, respectively, where these norms are defined as

$$\|H\|_p=\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|^p\,d\omega\right]^{1/p}$$

and

$$\|X\|_q=\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}|X(\omega)|^q\,d\omega\right]^{1/q}.$$

For example, with $H(\omega)$ chosen as unity, a consequence of (21) is that

$$|x(n)|\le\|X\|_q, \qquad \text{all } q\ge 1.$$

As another consequence, if we choose $p=1$, $q=\infty$, and use the fact that the L_∞ norm of $|X(\omega)|$ is the maximum value of $|X(\omega)|$ then we obtain the statement that

$$|y(n)|\le\max\left[|X(\omega)|\right]\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|\,d\omega.$$

As an alternative, with $p=2$, $q=2$,

$$|y(n)|\le\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|^2\,d\omega\right]^{1/2}\cdot\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}|X(\omega)|^2\,d\omega\right]^{1/2}.$$

To prevent overflow in the output we require that $|y(n)|<1$ and to insure this from (21) we will require that $\|H\|_p\|X\|_q<1$. Consequently, the input must be scaled in such a way that

$$\|X\|_q<1/\|H\|_p. \tag{22}$$

This condition is somewhat less general than (7) but in many cases is easier to apply. According to (22) with $p=2$, $q=2$, the condition is in terms of the energy in the input signal and the energy in the system impulse response. For $q=1$, $p=\infty$, (22) provides a bound in terms of the peak value of the magnitude of the transfer function, which is perhaps most appropriate for a sinusoidal input.

For the case of a random input (21) cannot be applied since the input and output do not have Fourier transforms. In this case the corresponding condition is phrased in terms of $\phi_{yy}(n)$ the autocorrelation function of the output, $\Phi_{xx}(\omega)$ the power density spectrum of the input, and $H(\omega)$ the magnitude of the system function. In particular, the inequality corresponding to (21) is

$$\phi_{yy}(n)\le\|H^2\|_p\|\Phi_{xx}\|_q \tag{23a}$$

or equivalently

$$\phi_{yy}(n)\le\|H\|_{2p}^2\|\Phi_{xx}\|_q. \tag{23b}$$

Since, if the input is zero mean, $\phi_{yy}(0)=\sigma_y^2$ it follows that

$$\sigma_y^2 \leq \|H\|_{2p}^2 \|\Phi_{xx}\|_q. \tag{24}$$

Two particular cases of interest are $p=1$, $q=\infty$ and $p=\infty$, $q=1$ so that

$$\sigma_y^2 \leq \|H\|_2^2 \|\Phi_{xx}\|_\infty \tag{25}$$

and

$$\sigma_y^2 \leq \|H\|_\infty^2 \|\Phi_{xx}\|_1. \tag{26}$$

As Jackson points out, (25) implies the most stringent condition on the input spectrum $\Phi_{xx}(\omega)$ whereas (26) implies the most stringent condition on the transfer function. From (25), if the input spectrum is white so that $\Phi_{xx}(\omega)=\sigma_x^2$ for all ω, then

$$\sigma_y^2 \leq \sigma_x^2 \|H\|_2^2 \tag{27}$$

with the input sequence Gaussian, then, the output will overflow no more often than the input overflows if

$$\|H\|_2 \leq 1. \tag{28}$$

More generally, (27) provides a basis for choosing the input variance to control the maximum percentage of time that the output can overflow.

D. Statistical Analysis of Roundoff Errors with Floating-Point Arithmetic

For the case of floating-point arithmetic, noise is introduced due both to the adds and the multiplies. In analyzing the effect of floating-point roundoff the effect of rounding will be represented multiplicatively so that if $[x]$ denotes rounding of the mantissa in a floating-point number, then

$$[x] = x(1+\epsilon). \tag{29}$$

To illustrate the analysis of roundoff errors with floating-point arithmetic let us consider a first-order filter. Let $w(n)$ denote the ideal response of the filter, that is, the response with no roundoff noise and let $y(n)$ denote the response of the filter in the presence of roundoff noise. Then following Liu and Kaneko [8] we can write that

$$w(n) = aw(n-1) + x(n) \tag{30}$$

$$y(n) = [ay(n-1)(1+\epsilon_n) + x(n)](1+\xi_n). \tag{31}$$

We assume that ϵ_n and ξ_n are uniformly distributed between -2^{-b} and 2^{-b}, are uncorrelated from iteration to iteration, are independent of each other, and also are independent of the signal. Letting $E(n)$ represent the error in the output, so that $E(n)=y(n)-w(n)$, we can write from the above two equations that

$$E(n) - aE(n-1) = aw(n-1)(\epsilon_n + \xi_n) + x(n)\xi_n = u(n) \tag{32}$$

where we have neglected second-order terms in ϵ, ξ, and E. Since ϵ and ξ are statistically independent of x, and of $w(n-1)$, the term $u(n)$ is easily shown to be a white-noise sequence. Its variance, of course, depends on the excitation $x(n)$. The derivation of (32) with the second-order terms neglected corresponds to representing the roundoff noise as an additive noise source that is statistically independent of the signal but whose variance depends on the signal variance. Specifically, consider the first-order network drawn in Fig. 5 with the two noise sources $e_1(n)$ and $e_2(n)$. From the model for multiplier

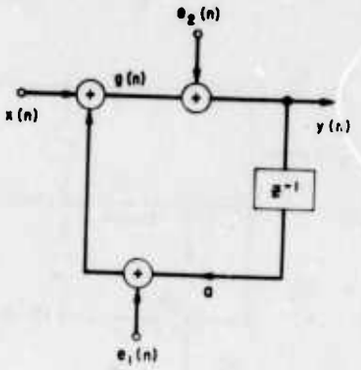


Fig. 5. Noisy first-order filter (floating point).

roundoff noise, the noise source $e_1(n)$ is given by

$$e_1(n) = ay(n-1)\epsilon_n \tag{33}$$

and the noise source $e_2(n)$ is given by

$$e_2(n) = g(n)\xi_n. \tag{34}$$

The analysis above in which we neglected second-order terms corresponds in this case to evaluating the variance of $e_1(n)$ and $e_2(n)$ by using the mean-square values for $y(n-1)$ and $g(n)$ that would result if no roundoff noise were present. Therefore, if we assume that $x(n)$ is a zero-mean white-noise input, with variance σ_x^2, then the variances of $e_1(n)$ and $e_2(n)$ are, respectively,

$$\sigma_{e_1}^2 = a^2\sigma_\epsilon^2\overline{y^2(n-1)} = a^2\sigma_\epsilon^2\sigma_x^2\frac{1}{1-a^2} \tag{35}$$

$$\sigma_{e_2}^2 = \sigma_\xi^2\overline{g^2(n)} = \sigma_\xi^2\sigma_x^2\frac{1}{1-a^2} \tag{36}$$

where the bar denotes expected value. Then, since $e_1(n)$ and $e_2(n)$ are independent, because ϵ_n and ξ_n are independent, the output noise variance is

$$\sigma_o^2 = \sigma_\epsilon^2\sigma_x^2\frac{1+a^2}{(1-a^2)^2} = \sigma_\epsilon^2\sigma_y^2\frac{1+a^2}{1-a^2} \tag{37a}$$

where we have assumed again that σ_ϵ^2 and σ_ξ^2 are equal. The output noise-to-signal ratio is

$$\frac{\sigma_o^2}{\sigma_y^2} = \sigma_\epsilon^2\frac{1+a^2}{1-a^2}. \tag{37b}$$

We can analyze the effect of roundoff noise in the second-order filter in a similar manner. In Fig. 6 is shown the network for a second-order filter with roundoff noise sources included. Note that since noise sources must be included due to addition, two summers are included to add the three variables in the feedback loop. The noise sources $e_3(n)$ and $e_4(n)$ represent the noise due to the multiplies and the noise sources $e_1(n)$ and $e_2(n)$ represent the noise due to the additions. With assumptions similar to those above in which we neglected second-order terms, we write that

$$\begin{aligned} e_1(n) &= y(n)\epsilon_1(n) \\ e_2(n) &= [y(n) - x(n)]\epsilon_2(n) \\ e_3(n) &= 2r\cos\theta y(n-1)\epsilon_3(n) \\ e_4(n) &= -r^2 y(n-2)\epsilon_4(n) \end{aligned} \tag{38}$$

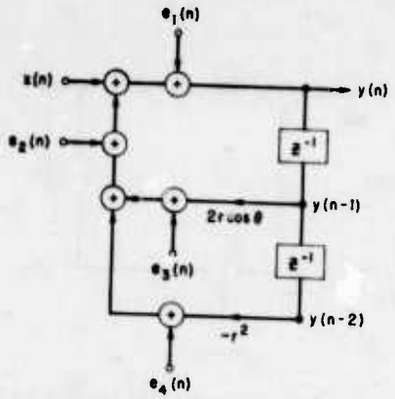


Fig. 6. Noisy second-order filter (floating point).

where ϵ_1, ϵ_2, ϵ_3, and ϵ_4 are independent random variables with equal variance σ_ϵ^2. If as before, $x(n)$ is assumed to be a white random process with variance σ_x^2, then the output noise-to-signal ratio for the second-order case is

$$\frac{\sigma_o^2}{\sigma_y^2} = \sigma_\epsilon^2\left[1 + G\left(3r^4 + 12r^2\cos^2\theta - 16\frac{r^4\cos^2\theta}{1+r^2}\right)\right] \quad (39)$$

where

$$G = \frac{1+r^2}{1-r^2}\left(\frac{1}{r^4 + 1 - 4r^2\cos^2\theta + 2r^2}\right). \quad (40)$$

For the high gain case, it is possible to compare fixed-point and floating-point arithmetic by approximating the expressions for the noise-to-signal ratio. For the first-order case, with $a = 1-\delta$, and $\delta \ll 1$, the result (37b) for the first-order filter, can be approximated as

$$\frac{\sigma_o^2}{\sigma_y^2} \simeq \frac{1}{3}\,2^{-2b}\frac{1}{\delta}. \quad (41)$$

Similarly for the second-order filter

$$\frac{\sigma_o^2}{\sigma_y^2} \simeq \frac{1}{3}\,2^{-2b}\left(\frac{3+4\cos^2\theta}{4\delta\sin^2\theta}\right) \quad (42)$$

where in (41) and (42) we have taken $\sigma_\epsilon^2 = (1/3)2^{-2b}$.

For fixed-point arithmetic we recall that for a white-noise input the noise-to-signal ratio behaved as $1/\delta^2$ and for a sinusoidal input as $1/\delta$. Comparison of (41) and (42) with (14) and (16) and (18) and (20) indicates a slightly larger noise-to-signal ratio for floating-point arithmetic as compared with fixed-point arithmetic with a sinusoidal input of known frequency but a significantly smaller noise-to-signal ratio for floating-point arithmetic as compared with fixed-point arithmetic with a white-noise input. It is important to keep in mind, however, that the noise-to-signal ratios for the fixed-point filters were computed on the basis that the input signal was as large as possible. If the input signal level decreases, the noise-to-signal ratio will increase since the output noise variance is independent of the input signal level. For floating-point arithmetic, on the other hand, the output noise variance is proportional to the output signal variance and as the input level is scaled up or down so is the roundoff noise. It is also important to note that the comparison just discussed assumes that the floating-point mantissa is equal in length to the entire fixed point word, and does not account for the extra bits needed for the characteristic. The authors [6] have previously compared fixed- and floating-point filters on the basis of equal total word length. However, in completing such a comparison one must take account of the large difference in hardware complexity between implementing floating-point arithmetic, and adding a few bits to a fixed-point arithmetic element.

Oppenheim [9] has proposed a realization of recursive digital filters using block floating-point arithmetic. Here the input and filter states (i.e., the inputs to the delay registers) are jointly normalized before the multiplications and additions are performed in fixed-point arithmetic. The scale factor (or exponent) obtained during the normalization is then applied to the final output to obtain a fixed-point output. The roundoff noise properties of such a realization were studied, and the noise-to-signal ratio was found to lie between that for fixed and floating point.

E. Zero-Input Limit Cycle Behavior of Digital Filters for Fixed-Point Arithmetic

In the preceding discussion the effect of arithmetic roundoff was modelled as an additive white-noise source, uncorrelated with the data. Justification of this model assumes that from iteration to iteration, the input can be expected to pass through several quantization levels. Consequently, this model is applied primarily when the input signal has a complicated behavior and cannot be expected to be valid in general. For example, consider a first-order filter for which the difference equation is

$$y_n = \alpha y_{n-1} + x_n \quad (43)$$

and for which the register length for the data is 4 bits and the coefficient α is 0.5. If the input x_n is 7/8 and if rounding is applied after the arithmetic then on successive iterations of the filter, the output will be:

$$y_0 = 7/8$$
$$y_1 = 1/2$$
$$y_2 = 1/4$$
$$y_3 = 1/8$$
$$y_n = 1/8, \qquad \text{for } n \geq 4.$$

Thus due to rounding, the output reaches a steady-state nonzero value and since the ideal steady-state output is zero, this nonzero value represents roundoff error. Clearly this kind of roundoff error cannot be modelled as white noise, but in fact represents a limit cycle due to the nonlinearity corresponding to the quantizer which implements the rounding. Limit cycle behavior of this type was first noted by Blackman [10] who referred to the amplitude intervals within which these limit cycles are confined as "deadbands." Blackman considered only first-order limit cycles corresponding to a dc behavior in the deadband. More generally, Jackson [11] has considered limit cycle behavior in first- and second-order filter sections with an analysis based on the location of the "effective" poles in the filter due to roundoff. Following the approach presented by Jackson, consider a first-order filter with a difference equation of the form of (43). Due to the register length constraint, the product αy_{n-1} must be rounded. Let $(\cdot)'$ denote the operation of rounding. If the register length is $(b+1)$ bits and if data are represented as fractions then

$$\left|(\alpha y_{n-1})' - \alpha y_{n-1}\right| \leq (\tfrac{1}{2})2^{-b}. \quad (44)$$

If y_{n-1} is such that $\left|(\alpha y_{n-1})'\right| = \left|y_{n-1}\right|$ then the magnitude of

the effective value of the coefficient is unity corresponding to the pole of the filter being on the unit circle. The range of values for which this condition is met is

$$|y_{n-1}| - |\alpha y_{n-1}| \leq (\tfrac{1}{2})2^{-b} \tag{45}$$

or

$$|y_{n-1}| \leq \frac{(0.5)2^{-b}}{1 - |\alpha|}. \tag{46}$$

This range of values is referred to as the deadband. Due to rounding, of course, values within the deadband must be in steps of 2^{-b}. For the first-order filter, when the filter state falls within this range and the input is zero, the effective pole is on the unit circle and the filter will support a limit cycle behavior. If the coefficient α is positive, as in the above example, the limit cycle response is dc, i.e., has constant magnitude and sign. For α negative the limit cycle behavior has constant magnitude but alternating sign.

For a second-order filter there is a larger variety of modes of limit cycle behavior. In particular, consider the second-order difference equation

$$y_n = x_n - \beta_1 y_{n-1} - \beta_2 y_{n-2}. \tag{47}$$

With $\beta_1^2 < +4\beta_2$ the filter poles occur as a complex conjugate pair and with $\beta_2 = 1$ the poles occur on the unit circle. The approach proposed by Jackson for examining the limit cycle behavior of the second-order filter corresponds to considering the filter behavior when the effect of rounding places the effective poles of the filter on the unit circle. With zero input the effective poles will be on the unit circle if

$$|y_{n-2}| - |(\beta_2 y_{n-2})| \leq \tfrac{1}{2}\, 2^{-b} \tag{48}$$

or

$$|y_{n-2}| \leq \frac{(0.5)2^{-b}}{1 - |\beta_2|}. \tag{49}$$

Thus if the output falls within this range the effective value of β_2 is unity so that the effective poles are on the unit circle. With the effective value of β_2 as unity, the effective value of β_1 controls the oscillation frequency.

A second mode of limit cycle behavior occurs in second-order filters when the effect of rounding is to place an effective pole at $z = \pm 1$. As shown by Jackson, the deadband corresponding to this mode is for values less than or equal to $1/(1 - |\beta_1| + \beta_2)$ in steps of integer multiples of 2^{-b}.

While this approach is somewhat heuristic, Jackson has found that these bounds are consistent with experimental results and hence he has hypothesized that they represent necessary and sufficient conditions. These bounds for second-order filters are summarized in Fig. 7, showing different deadband subregions in the β_1, β_2 plane. The number within an area in the β_1, β_2 plane represents the maximum magnitude of the limit cycle in multiples of 2^{-b} and the cross hatched region represents the region for which no limit cycles can occur.

Recently, Parker and Hess [12] have studied the limit cycle problem further, and found that these bounds are approximately correct and sufficient, but not necessary. In other words, there exist some limit cycles outside the regions specified by Fig. 7.

In addition to the above classes of limit cycles, a more severe type of limit cycle can occur due to overflow in filters implemented using one's-complement or two's-complement

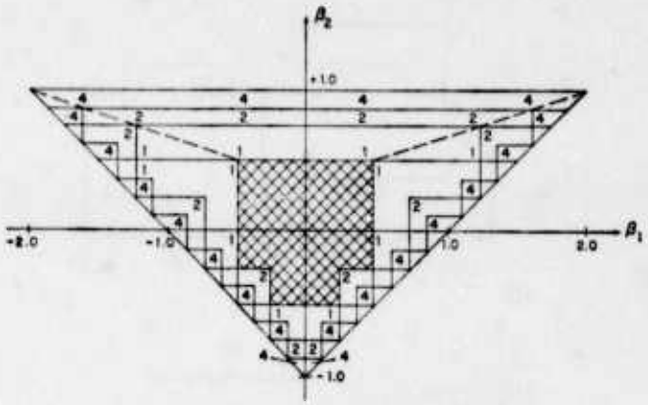


Fig. 7. Deadband subregions.

arithmetic. These limit cycles have been referred to as overflow oscillations [13] and can be avoided by using saturation arithmetic.

F. Effects of Parameter Quantization in Digital Filters

In the preceding sections we focussed on the effects of arithmetic roundoff in digital filters. Another consequence of the requirement of finite register length is that the filter coefficients cannot be specified exactly. Classical design procedures generally lead to filter coefficients with arbitrary accuracy and the implementation of the filter then requires that the coefficients be modified to fit the available register length. For hardware realizations of digital filters it is, of course, desirable to keep the register length as small as possible.

One common approach to the problem of parameter quantization is the use of filter configurations or structures which in some sense are least sensitive to inaccuracies in the parameters. One of the difficulties in evaluating the sensitivity of filter structures is the choice of a meaningful measure of the sensitivity. Most commonly, the sensitivity of the filter is tied to the movement of the poles of the filter. For this choice Kaiser [14] has shown that for a filter with clustered poles a cascade or parallel combination of first- and second-order sections provides more accuracy in the pole positions than a direct form realization. This is basically a consequence of the fact that for a polynomial whose roots are clustered, the sensitivity of the roots to changes in the polynomial coefficients increases as the order of the polynomial increases. Thus the roots can be more accurately controlled if the polynomial is factored into first- and second-order factors.

Even within the choice of first- and second-order sections some flexibility remains. For a direct form implementation of a pole pair as shown in Fig. 8(a) the coefficients are $-r^2$ and $2r\cos\theta$. For a given quantization on the coefficients the poles must lie on a grid in the z plane defined by the intersection of concentric circles, corresponding to quantization of r^2 and vertical lines, corresponding to quantization of $2r\cos\theta$. Such a grid is illustrated in Fig. 8(b). An alternative realization of a pole pair is the coupled form proposed by Rader and Gold [15], as shown in Fig. 9(a). In this case the coefficients are $r\cos\theta$ and $r\sin\theta$ and consequently the poles must lie on a rectangular grid as illustrated in Fig. 9(b). We note, for example, that for a given coefficient word length the direct form permits more accurate placement of poles with r close to unity and θ large while the coupled form is more advantageous for θ small. There are, in theory, many other structures in addition to the direct and coupled forms for implementing pole

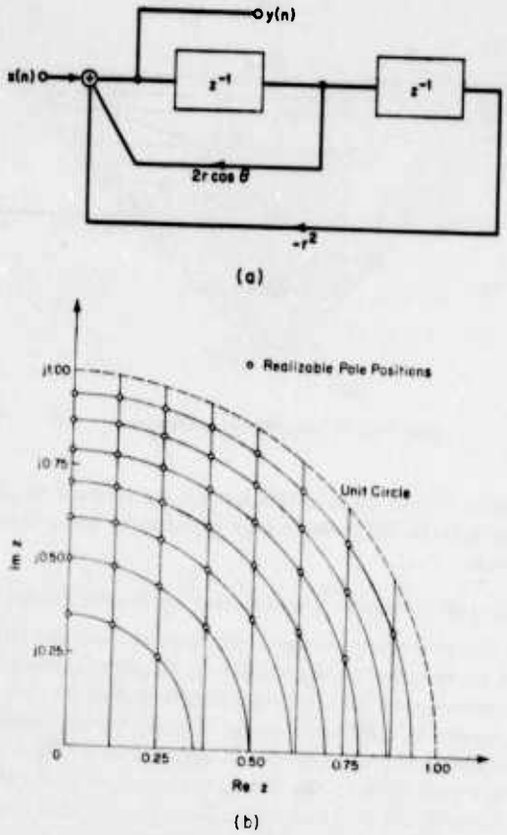


Fig. 8. (a) Direct form implementation of a pole pair. (b) Grid of allowable pole positions—direct form.

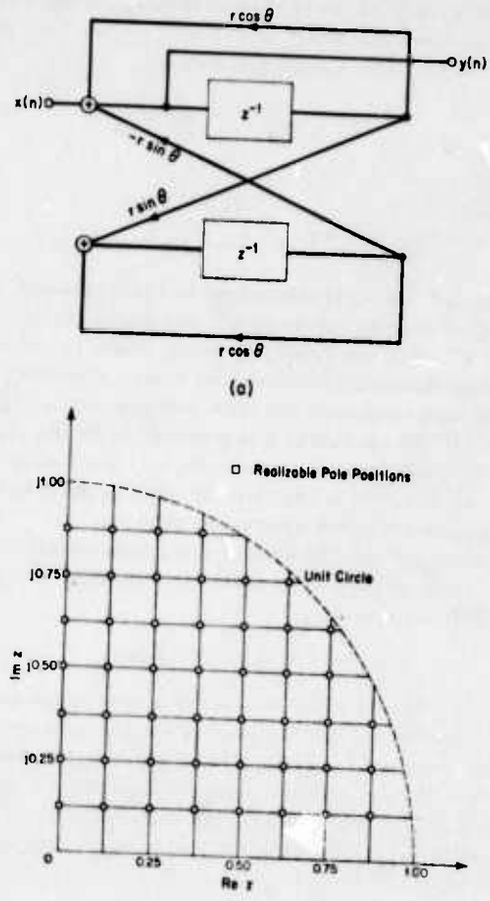


Fig. 9. (a) Coupled form implementation of a pole pair. (b) Grid of allowable pole positions—coupled form.

pairs although they are the most commonly considered [16]. Different structures, of course, imply different grids in the z plane and generally it is advantageous to choose a structure for which the grid is dense in the region of the z plane where the poles are to be located.

With a given choice of structure there remains the question as to how the pole locations on the grid should be chosen. A common procedure is to truncate or round the ideal coefficients. An alternative used by Avenhaus and Schussler [17] and also by Steiglitz [18] is to search over the grid in the vicinity of the ideal pole locations to select a grid point which locally minimizes the maximum error in the filter frequency response. As an alternative to the use of cascade or parallel connections of first- and second-order sections, more general filter structures can be considered. Digital wave filters, as proposed by Fettweis [19] and investigated by Bingham [20] and by Crochiere [21], appear to have much less sensitivity to parameter inaccuracies than the cascade form.

It would, of course, be desirable to incorporate the constraint of quantized coefficients into the design of digital filters. For nonrecursive filters, algorithmic design falls within the framework of integer linear programming. For recursive filters, however, the equations become nonlinear. In general, the development of design procedures with quantized coefficients remains an important area of research.

IV. Effects of Arithmetic Roundoff in the FFT

A. Introduction

The FFT algorithm [22] for computing the discrete Fourier transform (DFT) plays a central role in many signal processing applications [23]. As with the implementation of digital filters, it is important to understand the effect of finite register length arithmetic on the performance of the algorithm.

There are many forms of the FFT algorithm and the detailed effects of quantization will differ depending on the form used. The most commonly used forms of the algorithm are the radix-2 forms for which the size of the transform computed is an integer power of two. For the most part, the discussion below is phrased in terms of a particular form of the radix-2 FFT, commonly referred to as the decimation in time form of the algorithm; the results however are applicable with only minor modification to the decimation in frequency form. We feel that most of the ideas employed in the error analysis of the radix-2 forms of the algorithm can be utilized in other forms such as mixed radix, etc.

Our approach in analyzing noise in the FFT is basically statistical. In most cases, the predictions of the models are supported with experimental data (from Weinstein [24], unless otherwise stated). For floating and block floating point arithmetic, in order to simplify the analysis and obtain concrete results, it is convenient to assume a simple, white-noise model for the signal being transformed. Discussion of how the results might be expected to change for other types of signals is included, as are experimental noise measurements on FFT's of nonwhite signals.

B. The FFT Algorithm

The FFT algorithm is directed toward computing the DFT of a finite duration sequence $f(n)$, defined as

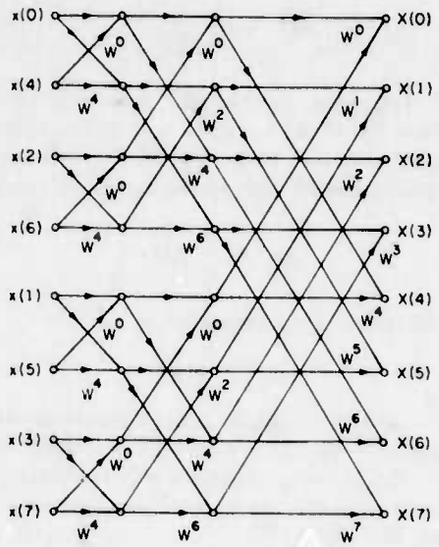


Fig. 10. FFT flow graph, $N=8$.

$$F(k) = \sum_{n=0}^{N-1} f(n) W^{nk}, \qquad W = e^{-j(2\pi/N)}. \tag{50}$$

A flow chart depicting the FFT algorithm for $N=8=2^3$ is shown in Fig. 10. A specific decimation in time algorithm is depicted. (An implementation of this particular form of the algorithm was used for the reported experimental work.) Some key aspects of this diagram, which are common to all standard radix-2 algorithms, are as follows. The DFT is computed in $\nu=\log_2 N$ stages. At each stage, the algorithm passes through the entire array of N complex numbers, two at a time, generating a new N number array. The νth array contains the desired DFT. The basic numerical computation operates on a pair of numbers in the $(m+1)$th array. This computation, referred to as a "butterfly" is

$$\begin{aligned} X_{m+1}(i) &= X_m(i) + \bar{W} X_m(j) \\ X_{m+1}(j) &= X_m(i) - \bar{W} X_m(j). \end{aligned} \tag{51}$$

Here, $X_m(i)$ and $X_m(j)$ represent a pair of numbers in the mth array, and $\bar{W}$ is some appropriate integer power of W, that is

$$\bar{W} = W^p = e^{-j2\pi p/N}. \tag{52}$$

The form of the butterfly computation is actually somewhat different for a decimation in frequency algorithm, where the computation is

$$\begin{aligned} X_{m+1}(i) &= X_m(i) + X_m(j) \\ X_{m+1}(j) &= [X_m(i) - X_m(j)]\bar{W}. \end{aligned} \tag{53}$$

At each stage, $N/2$ separate butterfly computations are carried out to produce the next array. The integer p varies with i, j, and m in a manner which depends on the specific form of the FFT algorithm that is used. Fortunately, our analysis is not tied to the specific way in which p varies. Also, the specific relationship between i, j, and m, which determines how we index through the mth array, is not important for the analysis. The details of the analysis for decimation in time and decimation in frequency differ somewhat due to the different butterfly forms, but the basic results for the dependence of noise-to-signal ratio on N do not change significantly. In our analysis we will assume a butterfly of the form (51), corresponding to decimation in time.

C. FFT Roundoff Noise with Fixed-Point Arithmetic

We will model the roundoff noise by associating an independent white-noise generator with each multiplier. This means that a noise source feeds into each node of the signal flow graph of Fig. 10 (excluding the initial array of nodes, since we are not considering A/D noise here). Since we are dealing with complex multiplications, these elemental noise sources are complex. Defining the complex variance σ_B^2 as the expected squared magnitude of such a noise source, we have

$$\sigma_B^2 = 4\,\frac{2^{-2b}}{12} \tag{54}$$

where it is assumed that each of the four real multiplications used to perform the complex multiplication is rounded separately. In Fig. 10, $3\times 8=24$ such noise sources must be inserted. To add the effects of each of the noise sources in evaluating the total roundoff noise in the output, we note that the transmission function from any node in the flow graph to any other connected node is multiplication by a complex constant of unity magnitude. Since we assume that all noise sources are uncorrelated, the noise variance at any output node is equal to σ_B^2 times the number of noise sources that propagate to that node. The general result which is easily verified for the case $N=8$ by inspection of Fig. 10 is that $(N-1)$ noise sources propagate to each output node so that the output noise variance σ_E^2 is given by

$$\sigma_E^2 = (N-1)\sigma_B^2$$

which for large N we take as

$$\sigma_E^2 \simeq N\sigma_B^2. \tag{55}$$

According to this result, the variance of the output noise is proportional to N, the number of points transformed. The effect of doubling N, or adding another stage in the FFT, is to double the output noise variance. Using the assumptions we have made thus far about the noise generators in the FFT (all uncorrelated, with equal variances), the output noise is white, i.e., the N noise samples $E(k)$ are mutually uncorrelated, with independent real and imaginary parts. This follows from the fact that the output of any butterfly is white (two outputs uncorrelated with equal variance, real and imaginary parts uncorrelated) if the input is white. Since the noise sources in our system are white, and all connected to the output via some combination of butterfly computations, the output noise must also be white.

In order to simplify the analysis leading to (55), we have neglected some details. First, we have associated equal variance noise sources with all multipliers, including where $W=1$ and j. In many programmed FFT's these multiplications are performed noiselessly. If we assume in the analysis that these multiplications are noiseless, the output noise variance will no longer be uniform over the output array. For example, the zeroth output point would be noiseless. The average variance over the output array will be somewhat lower than the result in (55), but will retain a linear dependence on N. Second, the assumption that all noise sources are uncorrelated is contradicted by the fact that the two noise sources associated with a given butterfly are negatives of each other, and therefore completely correlated. This does not affect the result for output noise variance, since the two outputs of a butterfly connect to a disjoint set of output points. However, it implies that the output noise samples $E(k)$ are somewhat correlated. These details are worth mentioning, but not worth analyzing here at

length, because they cloud the essential ideas of the analysis, are quite program-dependent, and do not change the essential character of the dependence of mean-squared output noise on N.

In implementing the FFT with fixed-point arithmetic we must insure against overflow. From (51) it follows that

$$\max\left[\left|X_m(i)\right|, \left|X_m(j)\right|\right] \leq \max\left[\left|X_{m+1}(i)\right|, \left|X_{m+1}(j)\right|\right] \quad (56)$$

and also that

$$\max\left[\left|X_{m+1}(i)\right|, \left|X_{m+1}(j)\right|\right] \leq 2\max\left[\left|X_m(i)\right|, \left|X_m(j)\right|\right]. \quad (57)$$

Equation (56) implies that the maximum modulus is nondecreasing from stage to stage so that, if the magnitude of the output of the FFT is less than unity then the magnitude of the points in each array must be less than unity,[1] i.e., there will be no overflow in any of the arrays.

In order to express this constraint as a bound on the input sequence, we note that the maximum possible output can be expressed in terms of the maximum input as

$$\left|X(k)\right|_{\max} \leq \left|x(n)\right|_{\max} \sum_{n=0}^{N-1} \left|W^{nk}\right| = N\left|x(n)\right|_{\max}. \quad (58)$$

Thus bounding the input sequence so that

$$\left|x(n)\right| < 1/N \quad (59)$$

will prevent overflow. To obtain an explicit expression for output signal variance, we assume $x(n)$ white, with real and imaginary parts each uniformly distributed in $(-1/\sqrt{2}N, 1/\sqrt{2}N)$. Then we have

$$\sigma_X^2 = \overline{\left|X(k)\right|^2} = N\sigma_x^2 = N\overline{\left|x(n)\right|^2} = \frac{1}{3N}. \quad (60)$$

Combining this with (55) yields

$$\frac{\sigma_E^2}{\sigma_X^2} = 3N^2\sigma_B^2. \quad (61)$$

The assumption of white input signal is not critical here. For example, if a complex sinusoid $x(n) = (1/N)\exp j(2\pi k_0 n/N + \phi)$ had been selected σ_E^2/σ_X^2 would still be proportional to N^2, which is the essential point of (61).

Equation (57) suggests an alternative procedure for preventing overflow. Since the maximum modulus increases by no more than a factor of two from stage to stage we can prevent overflow by requiring that $\left|x(n)\right| < 1$ and incorporating an attenuation factor of $1/2$ at each stage. Using this step-by-step scaling, the attainable output signal level (for white input) is the same as in (60) since the output signal level does not depend on where the scaling is done, but only on how much overall scaling is done. However, the output noise level will be much less than in (55) since the noise introduced at early stages of the FFT will be attenuated by the scaling which takes place at the later array. Quantitatively for $N = 2^r$

$$\sigma_E^2 = \sigma_{B'}^2 \sum_{k=1}^{N/2} \frac{1}{k} \quad (62)$$

where $\sigma_{B'}^2$ represents the roundoff noise introduced due to multiplication by W' and scaling and will consequently be slightly higher than σ_B^2. In particular, if we assume that the scaling is accomplished with rounding, it can be shown

$$\sigma_{B'}^2 = \frac{5}{6}2^{-2b}. \quad (63)$$

For large N, (62) is approximately

$$\sigma_E^2 = 2\sigma_{B'}^2 \quad (64)$$

and thus is much less than the noise variance resulting when all of the scaling is carried out on the input data.

Now, we can combine (64) with (60) to obtain the output noise-to-signal ratio for the case of step-by-step scaling and white input. We obtain

$$\frac{\sigma_E^2}{\sigma_X^2} = 6N\sigma_{B'}^2 = (5N)2^{-2b} \quad (65)$$

a result proportional to N, rather than to N^2. An interpretation of (65) is that the rms output noise-to-signal ratio increases as N, or by half a bit per stage. This result was first obtained by Welch [25]. It is important to note that the assumption of white signal is not essential in the analysis. The basic result of half-a-bit-per-stage increase holds for a broad class of signals, with only the constant multiplier in (65) being signal-dependent. In particular, for a general input with scaling at each array, the output variance is related to the variance of the input array by

$$\sigma_X^2 = \frac{1}{N}\sigma_x^2 = \frac{1}{N}\overline{\left|x(n)\right|^2} \quad (66)$$

so that

$$\frac{\sigma_E^2}{\sigma_X^2} = \frac{\frac{5}{3}N2^{-2b}}{\sigma_x^2} \quad (67)$$

where, to reduce noise-to-signal ratio, we would like to make σ_x^2 as large as possible but are limited by the constraint $\left|x(n)\right| < 1$. The result (67) has been verified experimentally for both wide-band and narrow-band signals [24], [25].

We should also note that the dominant factor causing the increase of σ_E^2/σ_X^2 with N is the decrease in signal level (required by the overflow constraint) as we pass from stage to stage. According to (63) and (64), very little noise (only a bit or two) is present in the final array. Most of the noise has been shifted off by the scalings. However, the mean-squared signal level has decreased by a factor of $1/N$ from its initial value, due to the scalings. Our output consists not of the DFT defined by (50) but of $1/N$ times this DFT.

We have assumed straight fixed point computation in this section, i.e., only preset attenuations were allowed, and we were not permitted to rescale on the basis of an overflow test. Clearly, if the hardware or programming facility are such that straight fixed point must be used, we should, if possible, incorporate attenuators of $1/2$ at each array rather than using a large attenuation of the input array.

[1] Actually one should discuss overflow in terms of the real and imaginary parts of the data, rather than the magnitude. However, $|x| < 1$ implies that $|\mathrm{Re}\,(x)| < 1$ and $|\mathrm{Im}\,(x)| < 1$, and only a slight increase in allowable signal level is achieved by scaling on the basis of Re and Im parts.

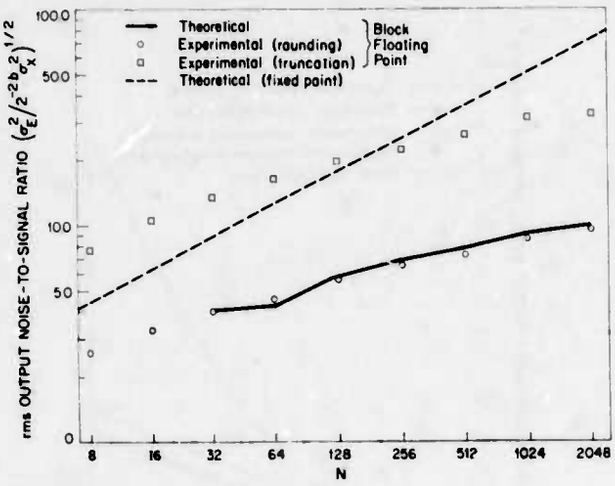


Fig. 11. Experimental and theoretical noise-to-signal ratios for block floating-point FFT.

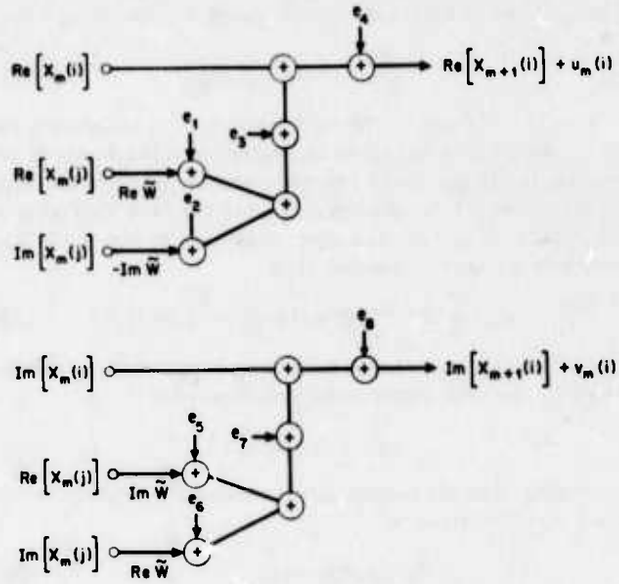


Fig. 12. Noisy butterfly computation (floating point).

A third approach to avoiding overflow is the use of block floating point. In this procedure the original array is normalized to the far left of the computer word, with the restriction that $|x(n)| < 1$: the computation proceeds in a fixed point manner, except that after every addition there is an overflow test; if overflow is detected, the entire array is shifted right 1 bit and the computation continues. The number of necessary shifts are counted to determine a scale factor or exponent for the entire final array. The output noise-to-signal ratio depends strongly on how many overflows occur, and at what stages of the FFT they occur. The positions and timing of overflows are determined by the signal being transformed, and thus, in order to analyze noise-to-signal ratio in block floating FFT, one needs to know the signal statistics. This is in contrast to the fixed point analysis above, where it was not necessary to assume specific signal statistics.

The necessary number of right shifts of the array is related to the peakiness of the DFT of the signal being transformed. If the constant signal $x(n) = 1$ or the single-frequency input $x(n) = \exp j(2\pi/N)k_o n$ is transformed, the output (with k_o an integer) will consist of a single nonzero point and (for $N = 2^\nu$) ν scalings of the array will be necessary, one for each stage.

A reasonable case to examine is the case of a white input signal; the DFT of a white signal is white, and one might expect (since the spectral energy is spread) that scalings at all stages would not be necessary, and a noise-to-signal ratio advantage over fixed point would be gained. This problem can be analyzed theoretically [24] but the analysis is quite involved and will be omitted. Instead, we will present some experimental results.

In Fig. 11 experimentally measured values of output noise-to-signal ratio are presented for block floating FFT's of white inputs, using rounded arithmetic. The quantity plotted is $(\sigma_E^2/2^{-2b}\sigma_x^2)^{1/2}$, the rms noise-to-signal ratio. For comparison, a theoretical curve representing fixed point noise-to-signal ratio (for rounded arithmetic) is also shown. We see that for white input block floating point provides some advantages over fixed point, especially for the larger transforms. For $N = 2048$, the rms noise-to-signal ratio for block floating point is about 1/8 that of fixed point, representing a 3-bit improvement.

An experimental investigation was used to examine how the results for block floating point change, when truncation rather than rounding is used. The results of this experiment are also shown in Fig. 11. Noise-to-signal ratios are generally a bit or two worse than for rounding. The rate of increase of noise-to-signal ratio with N seems to be about the same as for rounding.

D. FFT Roundoff Noise with Floating-Point Arithmetic

The effect of arithmetic roundoff with floating-point arithmetic has been analyzed theoretically and experimentally by Gentleman and Sande [26], by Weinstein [27], and by Kaneko and Liu [28]. As with the statistical analysis of roundoff errors with fixed-point arithmetic, noise is introduced due to each butterfly computation. As with floating-point errors in digital filters, we neglect second-order error terms so that noise sources are introduced after each multiplication and addition that are assumed to be white but for which the variance is proportional to the variance of the signal at that node. Unless the input signal is assumed to be white, the analysis becomes quite complicated due to the variation of the variance of the signal and therefore of the noise sources within each array. Kaneko and Liu have obtained detailed formulas for a general stochastic model of the input signal. We will confine attention here to the case of white input signal, where the signal at any array in the FFT is also white, with constant variance across the array.

In Fig. 12 a typical butterfly computation (only top half) is indicated, including the noise sources due to multiplication and addition. The assumption of white input signal implies that

$$\overline{[\mathrm{Re}\,(X_m)]^2} = \overline{[\mathrm{Im}\,(X_m)]^2} = \overline{\tfrac{1}{2}\,|X_m|^2} \tag{68}$$

and application of our floating-point noise model as in Section III-D yields the noise source variances

$$\sigma_{e_1}^2 + \sigma_{e_2}^2 = \sigma_{e_5}^2 + \sigma_{e_6}^2 = \sigma_{e_3}^2 = \sigma_{e_7}^2 = \tfrac{1}{2}\,\sigma_e^2\,\overline{|X_m|^2} \tag{69}$$

$$\sigma_{e_4}^2 = \sigma_{e_8}^2 = \sigma_e^2\,\overline{|X_m|^2}. \tag{70}$$

The variance of the complex noise source $U_m = u_m + jv_m$ is then

$$\overline{|U_m|^2} = 4\sigma_\epsilon^2 \overline{|X_m|^2} \tag{71}$$

so that the variance of the noise generated in computing the $(m+1)$th array is $4\sigma_\epsilon^2$ times the variance of the signal in the mth array. If the input (zeroth) array is white noise with variance σ_x^2 then the noise generated in the $(m+1)$th array is $2^m\sigma_x^2(4\sigma_\epsilon^2)$. If σ_{om}^2 is the output noise due to the noise generated in the $(m+1)$th array, then

$$\sigma_{om}^2 = 2^{\nu-(m+1)}2^m\sigma_x^2(4\sigma_\epsilon^2) = 2N\sigma_\epsilon^2\sigma_x^2. \tag{72}$$

Since the noise generated in each array is assumed to be independent, the total output noise variance σ_E^2 is

$$\sigma_E^2 = 2\nu N\sigma_\epsilon^2\sigma_x^2. \tag{73}$$

By noting that the output signal variance is related to the input signal variance by

$$\sigma_X^2 = N\sigma_x^2 \tag{74}$$

the result follows:

$$\frac{\sigma_E^2}{\sigma_X^2} = 2\sigma_\epsilon^2\nu. \tag{75}$$

A further result, which can be derived from our model, is an expression for the final expected output noise-to-signal ratio which results after performing an FFT and an inverse FFT on a white signal $x(n)$. The inverse FFT introduces just as much roundoff noise as the FFT itself, and thus the resulting output noise-to-signal ratio is

$$\frac{\sigma_E^2}{\sigma_x^2} = 4\sigma_\epsilon^2\nu \tag{76}$$

or just double the result in (75).

In order to see the implications of (75) or (76) in terms of register length requirements, it is useful to express these results in units of bits. We use

$$(\sigma_E^2/\sigma_X^2\sigma_\epsilon^2) \text{ bits} = \tfrac{1}{2}\log_2(2\nu) \tag{77}$$

to represent the number of bits by which the rms noise-to-signal ratio increases in passing through a floating point FFT. For example, for $\nu = 8$ this represents 2 bits and for $\nu = 11$ it represents 2.23 bits. The number of bits of rms noise-to-signal ratio increases as $\log_2(\log_2 N)$, so that doubling the number of points in the FFT produces a very mild increase in output noise, significantly less than the half-bit-per-stage increase for fixed-point computation. In fact, to obtain a half-bit increase in the result above, we would have to double ν, or square N.

In the analysis leading to (75), we have not considered the fact that multiplications by 1 can be performed noiselessly. For a specified radix-2 algorithm, such as the decimation in time algorithm shown in Fig. 10, these reduced variances for $W = 1$ and j can be included in the model to obtain a slightly reduced prediction for output noise-to-signal ratio. However, for reasonably large N, this modified noise analysis yields only slightly better predictions of output noise than does the simplified analysis above.

A consequence of our analysis leading to (75) is that the output noise is white. This follows from the fact that each array of noise sources is white. The reduced noise source variance for $W = 1$ and j implies that for some arrays there will be a variation of noise source variance over the array. This implies a slight variation of output noise variance over the output array, and thus our modified noise analysis will only predict an average noise variance over the output array.

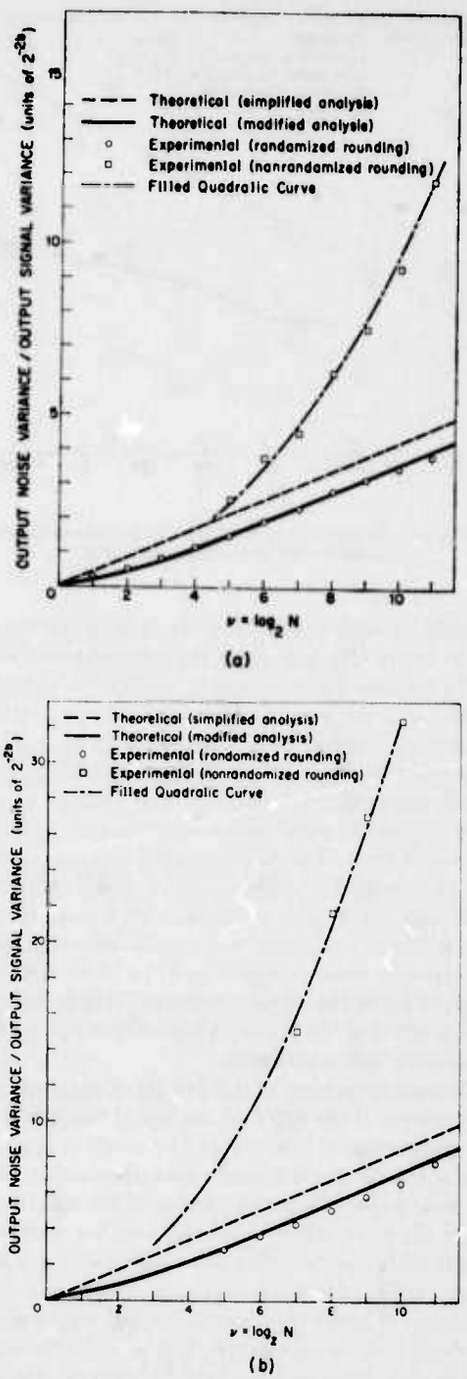


Fig. 13. (a) Experimental and theoretical noise-to-signal ratios for floating point FFT. (b) Experimental and theoretical noise-to-signal ratios for floating-point FFT and inverse.

The results discussed above have been verified with excellent agreement as shown in Fig. 13(a) and (b). To obtain this agreement, however, it was necessary to use randomized rounding, i.e., randomly rounding up or down when the value of mantissa was exactly $(1/2)2^{-b}$. The modified theoretical

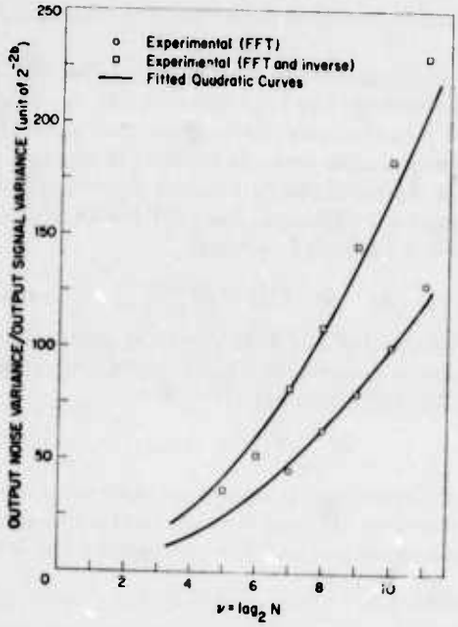


Fig. 14. Experimental noise-to-signal ratios for floating point FFT and FFT-inverse FFT; truncation used instead of rounding.

curve shown was obtained by taking into account reduced noise source variances for $W = 1$ and $W = j$. Also shown are experimental results for nonrandomized rounding. These results were fitted empirically with a curve of the form $a\nu^2$, but this quadratic dependence was not established theoretically. Noise-to-signal ratios were also measured for the case where truncation rather than rounding was used in the arithmetic; the results, with empirically fitted quadratic curves, are shown in Fig. 14.

Our analysis, and all the above experiments, applied to the case of white signal. Some experimental investigation has been carried out as to whether the predictions are valid when the signal is nonwhite. Specifically, the noise introduced in computing an FFT was measured for sinusoidal signals of several frequencies, for $\nu = 8$, 9, 10, and 11. The results, averaged over the input frequencies used, were within 15 percent of those predicted by (75). In these experiments, the "randomized" rounding procedure was used.

E. Effects of Coefficient Quantization in the FFT

As with the implementation of digital filters, the implementation of the FFT algorithm requires the use of quantized coefficients. While a completely definitive study of the effects of coefficient quantization in the FFT remains to be done, two approaches have been pursued for which some results have been obtained.

Although the nature of coefficient quantization is inherently nonstatistical, Weinstein [24] has obtained some useful results by means of a rough statistical analysis. This statistical analysis corresponds to introducing random jitter in the coefficients and determining the output noise-to-signal ratio due to this noise. While the detailed effect due to coefficient error due to quantization is different than that due to jitter, it is reasonable to expect that in a gross sense the magnitude of the errors is comparable.

To develop this statistical analysis, we let $F(k)$ denote the DFT of a sequence $f(n)$ and $\hat{F}(k)$ the result of transforming $f(n)$ with a radix-2 FFT algorithm with jittered coefficients. Then

$$F(k) = \sum_{n=0}^{N-1} f(n) W^{nk} \tag{78}$$

and

$$\hat{F}(k) = \sum_{n=0}^{N-1} f(n) \Omega_{nk}. \tag{79}$$

Because of the form of the FFT algorithm each element Ω_{nk} will be a product of $\nu = \log_2 N$ quantized coefficients. Thus

$$\Omega_{nk} = \prod_{i=1}^{\nu} (W^{a_i} + \delta_i) \tag{80}$$

where

$$\prod_{i=1}^{\nu} W^{a_i} = W^{nk} \tag{81}$$

with b bits for the real and imaginary parts of each of the coefficients, excluding sign, $|\delta_i|$ is less than or equal to $(\sqrt{2})2^{-b}$. If we assume that the real and imaginary parts of the jitter in the coefficients are uncorrelated and uniformly distributed between plus and minus $(1/2)2^{-b}$ then σ_δ^2, the variance of δ_i is $\sigma_\delta^2 = 2^{-2b}/6$. The error in the computation of the DFT can be expressed as

$$E(k) = \hat{F}(k) - F(k) = \sum_{n=0}^{N-1} f(n)(\Omega_{nk} - W^{nk}). \tag{82}$$

From (80) and (81) we can express the factor $(\Omega_{nk} - W^{nk})$ as

$$(\Omega_{nk} - W^{nk}) = \sum_{i=1}^{\nu} \delta_i \prod_{\substack{j=1 \\ j \neq i}}^{\nu} W^{a_i} + \text{higher order terms.} \tag{83}$$

If we neglect higher order error terms, and assume that δ_i are mutually uncorrelated then the variance of $(\Omega_{nk} - W^{nk})$ is equal to $\nu(2^{-2b}/6)$. Finally, assuming that all elements Ω_{nk} are uncorrelated with each other and with the input signal, the output error variance σ_E^2 is

$$\sigma_E^2 = \nu \frac{2^{-2b}}{6} \sum_{n=0}^{N-1} |f(n)|^2. \tag{84}$$

Since from Parseval's relation

$$\sum_{n=0}^{N-1} |f(n)|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |F(k)|^2 = \text{mean-squared output signal} \tag{85}$$

the ratio of mean-squared output error to mean-squared output signal is thus

$$\sigma_E^2 \Big/ \left[\left(\frac{1}{N}\right)\left(\sum_{n=0}^{N-1} |F(k)|^2\right)\right] = \left(\frac{\nu}{6}\right) 2^{-2b}. \tag{86}$$

Although we would not expect (86) to predict with great accuracy the error in an FFT due to coefficient quantization, it is helpful as a rough estimate of the error. The key result of (86), which we would like to test experimentally, is that the error-to-signal ratio increases very mildly with N, being proportional to $\nu = \log_2 N$, so that doubling N produces only a slight increase in the error-to-signal ratio.

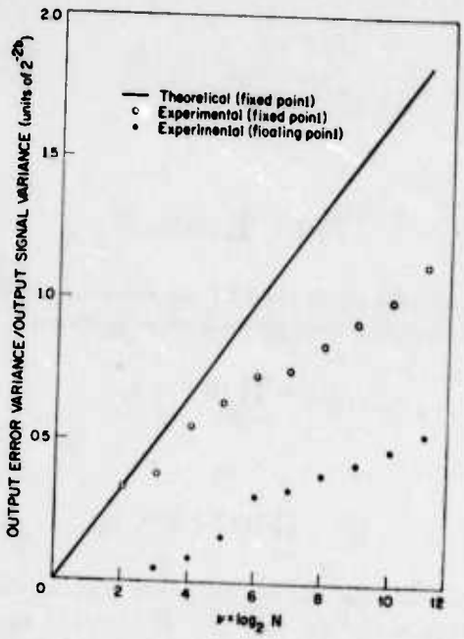


Fig. 15. Errors due to coefficient quantization in FFT.

To test this result, experimental measurements on errors due to coefficient quantization were made. In each run, a sequence $f(n)$—white in the sense that all $2N$ real numbers making up the N-point complex sequence were mutually uncorrelated, with zero means, and equal variances—was obtained using a random number generator. This sequence was transformed twice, once using a 36-bit coefficient table, and once using a coefficient table rounded to much shorter word length (e.g., 12 bits). For each transform, 36-bit accuracy was used in the arithmetic to make the effect of roundoff error negligible. The results were subtracted, squared, averaged over the output array, and divided by the output signal variance (N times the input signal variance) to obtain an experimental output error-to-signal ratio. For each value of N, several random sequences were transformed and the results were averaged to obtain statistically convergent estimates.

These results are displayed in Fig. 15; the quantity plotted is $\sigma_E^2/2^{-2b}\sigma_F^2$ where σ_F^2 is the mean-squared output signal as defined in (85). The theoretical curve corresponding to (86) is shown, and the circles represent measured output error-to-signal ratio for the fixed-point case. We note that the experimental results generally lie below the theoretical curve. No experimental result differs by as much as a factor of two from the theoretical result, and since a factor of two in σ_E^2/σ_F^2 corresponds to only half-a-bit difference in the rms output error, it seems that (86) is a reasonably accurate estimate of the effect of coefficient errors. The experimental results do seem to increase essentially linearly with ν, but with smaller slope than given in (86).

In the above, fixed-point arithmetic has been assumed. However, since a block floating-point FFT will generally use fixed-point coefficients, our results are valid for the block floating-point case also. With some slight modifications, it is possible to obtain similar results for the floating-point case. Except for a constant factor, the floating- and fixed-point results are the same. Experimental results for the floating-point case are represented by the solid dots in Fig. 15, and are observed to be slightly lower than the results for the fixed-point case.

A different approach to the characterization of FFT coefficient quantization has been taken by Tufts, Hersey, and Mosier [29]. In their analysis the effect of coefficient quantization is represented in terms of the level of spurious sidelobes introduced. In particular, a sequence $f(n) = u_o(n - n_o)$ where $u_o(n)$ denotes a unit sample, has a DFT with a purely sinusoidal real and imaginary part, i.e.,

$$F(k) = W^{n_o k} \tag{87}$$

and the inverse DFT of $F(k)$ should, of course, have only a single nonzero component. Due to coefficient quantization, however, the DFT obtained is

$$\hat{F}(k) = \Omega_{n_o k} \tag{88}$$

and since the real and imaginary parts are not exactly sinusoidal, the inverse DFT, with exact coefficients, of $F(k)$ will have spurious components. For each of the set of N sequences

$$f_k(n) = u_o(n - n_k), \qquad k = 0, 1, \cdots, N-1. \tag{89}$$

Tufts *et al.* compute the DFT with quantized coefficients followed by the inverse DFT with accurate coefficients. At the output of the inverse DFT, the size and frequency locations of the spurious sidelobe components produced due to the quantized coefficients are observed. Since any function $f(n)$ can be constructed as a weighted sum

$$f(n) = \sum_{k=0}^{N-1} a_k f_k(n) \tag{90}$$

the spurious sidelobes produced for any $f(n)$ can in principle be determined by combining the responses due to a set of impulses. But carrying out such a combination is not practical for arbitrary $f(n)$. Tufts *et al.* have, however, tabulated the worse sidelobe levels encountered for any $f_k(n)$ as a function of the number of bits retained in the coefficients, for the case of a 64-point FFT and sign-magnitude representation of coefficients.

V. Examples

A. Introduction

In the preceding sections the effects of arithmetic roundoff have been analyzed for simple (first- and second-order) digital filters and the FFT. These algorithms are the basic building blocks in more complicated digital processing such as a higher order digital filter or a convolutional filter realized via the FFT. Examples will be presented in this section to indicate how some of the ideas developed above can be applied to analyze and to choose the most advantageous configuration for such systems. The first two examples concern the realization of higher order recursive filters and have borrowed from the work of other authors. The third example deals with an FFT filter.

B. Fixed-Point Digital Filter in Cascade and Parallel Form

After a digital filter has been specified in terms of its poles and zeroes, and the type of arithmetic has been selected, a choice must still be made among the various possible configurations of the filter which will differ with respect to the effects of roundoff noise. An exhaustive study of the selection

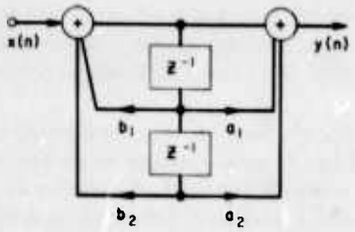


Fig. 16. Second-order section with poles preceding zeros. Used in Jackson's $1P$ parallel form with $a_2=0$. Also used in $1D$ direct form.

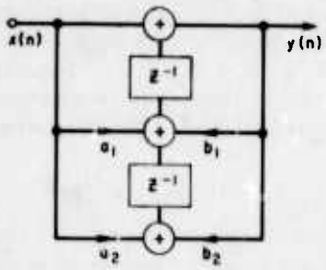


Fig. 17. Second-order section with zeros preceding poles. Used in $2P$ parallel form with $a_2=0$. Also used in $2D$ direct form.

of filter form is beyond the scope of this paper, but an excellent example of the necessary considerations is given by Jackson [30] in his analysis of roundoff noise for fixed-point digital filters realized in cascade and parallel form

Jackson considers two parallel form realizations: the $1P$ form where the individual second-order sections are realized, as shown in Fig. 16, with the poles preceding the zeros, and the $2P$ form where zeros precede poles in the individual sections, as shown in Fig. 17. (Figs. 16 and 17 do not show all the scaling coefficients needed to prevent overflow.) His analysis indicates that for a variety of scaling criteria (based on different L_p norms[2] of the input signal) and for various measures of the output noise (such as its total power, or its peak spectral value), the output signal-to-noise ratios of the two forms are very close. Generally, a very slight advantage will be gained with the $1P$ form.

Comparison of the two parallel forms basically reduces to a comparison of the noise properties of the two forms of second-order sections, since the noises from the second-order sections are simply additive in the output of the parallel form. Hence the discussion above applies to a comparison of second-order section realizations. Another form of second-order section which could be considered is a coupled form as shown in Fig. 18. For the case $a_1=r\cos\theta$, $a_2=r\sin\theta$, this filter has poles at $z=re^{\pm j\theta}$ and a zero at $z=r\cos\theta$. The coupled form noise-to-signal ratio has been compared [24] to ratios for forms essentially the same as those in Figs. 16 and 17 for the

[2] If

$$F(\omega)=\sum_{n=-\infty}^{\infty} f(n)\exp\left(-j2\pi\frac{\omega}{\omega_s}n\right)$$

represents the Fourier transform of a signal or of a filter impulse response $f(n)$, then the corresponding L_p norm is

$$\|F\|_p=\left(\frac{1}{\omega_s}\int_0^{\omega_s}|F(\omega)|^p\,d\omega\right)^{1/p}$$

where ω_s denotes sampling frequency.

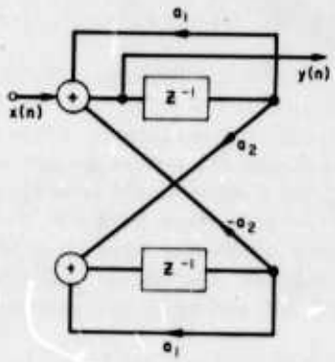


Fig. 18. Coupled form for second-order section.

case of white-noise input and an absolute overflow constraint (through the type of analysis given in Section III-C). The coupled form was found to have substantially lower noise-to-signal ratio for filters with high gain and low resonant frequencies. For $\delta=1-r$, and $\delta\ll 1$, the results vary as

$$\frac{\sigma_e^2}{\sigma_y^2}\sim 1/\delta^2\sin^2\theta \qquad \text{(forms of Figs. 16 and 17)}$$

$$\frac{\sigma_e^2}{\sigma_y^2}\sim 1/\delta^2 \qquad \text{(coupled form).} \qquad (91)$$

The implication of this result, together with the somewhat reduced coefficient sensitivity for the coupled form, is that this form may be a good choice in some situations, despite the fact that its implementation requires four multiplications instead of three for a pole pair and a single zero.

As stated above, Jackson found not much difference between the noise properties of the $1P$ and $2P$ parallel forms. However, the situation is more interesting in the case of the cascade form. Here he finds that large differences are possible between the roundoff noise outputs of the $1D$ (poles before zeros in individual sections) and $2D$ (zeros before poles in each section) forms. Also the ordering of the sections and the pairing of poles and zeros have important effect on the output signal-to-noise ratio. Jackson's analyses lead to several rules of thumb for selection of $1D$ or $2D$, for ordering of sections, and for pairing of poles and zeros.

In general, the choice of configuration depends on which L_p norm of the scaled transfer functions is constrained to prevent overflow and on which L_r norm of the output noise spectrum is used as a measure of performance. Two L_p constraints on the filter are of particular interest: the $p=\infty$ case, where the peak value of the transfer function to each possible overflow node is constrained; and the $p=2$ case where the rms transfer function to each node is constrained. The choice $p=\infty$ is just slightly less stringent than the absolute overflow constraint (7), and prevents overflow even when the input is a narrow-band signal at resonance of the relevant transfer function. The $p=2$ constraint is more appropriate for preventing overflow when the input is wide-band in nature. Two L_r norms on the output noise spectrum are of particular interest: the $r=1$ norm which measures the total output noise power, and the $r=\infty$ norm which measures the peak value of the spectrum of the output noise.

With regard to selection of $1D$ or $2D$ forms, Jackson's rule of thumb says to select $1D$ when $p=2$, $r=\infty$ and $2D$ when

$p=\infty$, $r=1$; for $p=2$, $r=1$ and for $p=\infty$, $r=\infty$, either form may be selected.

The choice of ordering of sections also depends on the norms which are selected. For $p=2$, $r=\infty$, the sections should be ordered in decreasing peakedness, where peakedness is defined as the peak gain of a section divided by its rms gain. For $p=\infty$, $r=1$, the sections should be ordered in increasing peakedness. For $p=2$, $r=1$ and for $p=\infty$, $r=\infty$, the choice of ordering depends on whether form 2*D* or 1*D* is chosen for the individual sections. Decreasing peakedness should be chosen with form 2*D*, and increasing peakedness with form 1*D*.

The rule for pairing of poles and zeros is as follows: Let $|H_n(\omega)|$ denote the magnitude of the frequency response of the nth section, and M_n denote the maximum over ω of $|H_n(\omega)|$. Then the pairing should be chosen such that the maximum over n of M_n is minimized.

The above rules are illustrated by Jackson with a specific filter example—a sixth-order Chebyshev band rejection filter, and the results are in accord with his rules. He analyzes the output noise of this filter for parallel forms 1*P* and 2*P* and for all orderings of cascade forms 1*D* and 2*D* (with proper pole–zero pairing). Little difference is seen between the two parallel forms. For $p=2$, $r=\infty$ the peak output noise spectrum is 7–12 dB worse for the 2*D* cascade forms than for the 1*D* forms; while for $p=\infty$, $r=1$, the output noise power is 7–12 dB worse for 1*D* than for 2*D*. The effects of pole–zero pairing and of ordering of sections also follow quite well the rules previously stated. The parallel forms turn out to be slightly superior to the best cascade forms with respect to roundoff noise.

As Jackson indicates, these rules of thumb have certain qualifications and are not always valid. However, they have been shown to be helpful in a variety of types of examples [31].

C. Choice of Form for Floating-Point Digital Filter

By means of an example, Liu and Kaneko [8] have compared the direct, cascade, and parallel form realizations of a floating-point digital filter. The filter selected was an eighth-order low-pass elliptic filter. The noise-to-signal ratio for the parallel form was about 20 dB worse than for the direct form, while the cascade form was comparable to (about 1.5 dB worse than) the parallel form.

Various orderings of cascade form floating-point filters have not been studied in detail. Probably floating-point cascade filters are not too sensitive to ordering since large variations in signal level from stage to stage can be accommodated by the floating-point exponent.

A comparison of the noise-to-signal ratio properties of floating-point second-order sections where poles precede zeros (Fig. 16) and where zeros precede poles (Fig. 17) indicates that at least for white-noise inputs the behavior of the two forms is essentially identical. For a high-gain second-order section of low resonant frequency, a coupled form realization yields some noise-to-signal ratio advantage over both of these two forms.

D. FFT Filter

The results of our roundoff noise analysis for fixed-point FFT will now be applied to obtain an expression for the output noise-to-signal ratio of a finite impulse response digital filter, implemented by means of the FFT. The overflow constraints of this type of filter will be accounted for in the analysis. Attention will be focussed on a prototype low-pass filter with 256-point impulse response and a cutoff frequency of 1/4 the half sampling frequency. Rounded arithmetic will be assumed.

Let us examine the basic steps in the filtering computation, tracing the buildup of noise variance as we proceed. First the FFT is used to compute the DFT of a section of input. In the implementation of a filter with 256-point impulse response, it is reasonable to compute a 512-point FFT, where the input consists of 256 data samples and 256 zeros. Actually 512 real input samples would be treated simultaneously, by placing sections of 256 real samples in both the real and imaginary parts of the input to the FFT. To guarantee against overflow, a scaling of 1/2 is needed at each stage of the FFT yielding an overall attenuation of 1/512. The samples of the input sequence must be less than unity in magnitude. The noise $E_1(k)$ at the output of this first DFT has variance

$$\sigma_{E_1}^2 = \overline{|E_1(k)|^2} = \frac{5}{3}2^{-2b}. \tag{92}$$

This noise variance is small, because most of the roundoff noise has been shifted off by the attenuations. However, the scalings have also caused the mean-squared signal to decrease by a factor of 1/512.

Next this computed transform is multiplied by a sequence $H(k)$ representing the DFT of a 512-point sequence $h(n)$ consisting of the filter impulse response plus 256 zeros. This complex multiplication introduces roundoff noise of variance $2^{-2b}/3$. Assuming that we have chosen $|H(k)|<1$, the mean square of the noise E_1 becomes reduced by

$$\frac{1}{512}\sum_{k=0}^{511}|H(k)|^2 = B \tag{93}$$

a ratio of the filter bandwidth to the sampling frequency. Thus after the multiplication, the variance of the total noise $E_2(k)$ is

$$\sigma_{E_2}^2 = \frac{2^{-2b}}{3} + B\frac{5}{3}2^{-2b}. \tag{94}$$

This noise is not white, but has a component whose spectrum has been shaped by the filter.

For the example under consideration $B\approx 1/4$, so

$$\sigma_{E_2}^2 \approx \frac{3}{4}2^{-2b}. \tag{95}$$

Note that $\sigma_{E_2}^2$ is slightly less than $\sigma_{E_1}^2$ and represents only about a bit of noise. However, if the signal spectrum is flat, the mean-squared signal will also be reduced somewhat due to the multiplication by $H(k)$.

Now an inverse transform is computed to obtain a section of output. The noise variance at the output of this transform depends on how many scalings are necessary in the inverse FFT. In order to determine how many scalings are necessary, a bound on the output of the circular convolution is required [32]. For a particular filter, such a bound can be stated as

$$|y(n)| < \sum_{l=0}^{M-1}|h(l)| \tag{96}$$

where $y(n)$ is the output and M is the length of the impulse

response. The prototype filter has an impulse response

$$h(n) = \frac{2}{256}\left[\frac{1}{2} + \sum_{k=0}^{31}(-1)^k \cos\frac{2\pi kn}{256} + 0.7\cos\frac{2\pi(32)n}{256} - 0.225\cos\frac{2\pi(33)n}{256} + 0.01995\cos\frac{2\pi(33)n}{256}\right] \quad (97)$$

and

$$\sum_{n=0}^{255} |h(n)| = 3.12. \quad (98)$$

Hence, only two scalings (at the first two arrays) are necessary in the inverse transform. Then, in propagating through the IFFT (inverse FFT), the variance of the noise $E_2(k)$ increases by a factor of 512/16. (The 512 represents the gain of the inverse DFT, and the 1/16 is due to the scalings.) The variance of the additional output noise $E_3(k)$ caused by roundoff in the IFFT can be estimated easily via the method of Section IV-C. The result is

$$\sigma_{E_3}^2 = \sigma_{B'}^2\left(\frac{512}{8} + \frac{512}{4}\right) + \sigma_B^2\sum_{k=1}^{6} 2^k = (202)2^{-2b}. \quad (99)$$

The total mean-squared output noise is

$$\sigma_E^2 = \frac{512}{16}\sigma_{E_2}^2 + \sigma_{E_3}^2 = (226)2^{-2b} \quad (100)$$

or in units of bits of rms output noise

$$\tfrac{1}{2}\log_2(2^{2b}\sigma_E^2) = 3.91 \text{ bits}. \quad (101)$$

The mean-squared output signal can be estimated if specific statistics are assumed for the input signal $x(n)$. As an example, assume that $x(n)$ is white with variance $\sigma_x^2 = 2/3$. This variance goes through an attenuation of 1/512 in the first FFT, an attenuation of $B = 1/4$ due to multiplication by $H(k)$, and a gain of 512/16 in the inverse transform. The mean-squared output signal is then

$$\sigma_y^2 = \left(\frac{1}{512}\right)\left(\frac{1}{4}\right)\left(\frac{512}{16}\right)\left(\frac{2}{3}\right) = \frac{1}{96} \quad (102)$$

and the output noise-to-signal ratio is

$$\frac{\sigma_k^2}{\sigma_y^2} \approx (22\,000)2^{-2b}. \quad (103)$$

Assuming an input noise-to-signal ratio (due to A/D noise) of $(1/4)2^{-2b}$, the noise-to-signal ratio has worsened by a factor of about 5500, or

$$\tfrac{1}{2}\log_2 5500 = 6.15 \text{ bits} \quad (104)$$

in passing through the FFT filter.

References

[1] W. R. Bennett, "Spectra of quantized signals," *Bell Syst. Tech. J.*, vol. 27, pp. 446–472, 1948.

[2] B. Widrow, "Statistical analysis of amplitude-quantized sampled-data systems," *AIEE Trans. (Appl. Indust.)*, vol. 81, pp. 555–568, Jan. 1961.

[3] B. Liu, "Effect of finite word length on the accuracy of digital filters—A review," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 670–677, Nov. 1971.

[4] J. B. Knowles and R. Edwards, "Effect of a finite-word-length computer in a sampled-data feedback system," *Proc. Inst. Elec. Eng.*, vol. 112, pp. 1197–1207, 1965.

[5] B. Gold and C. M. Rader, "Effect of quantization noise in digital filters," in *Proc. Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 28, pp. 213–219, 1966.

[6] C. Weinstein and A. V. Oppenheim, "A comparison of roundoff noise in floating point and fixed point digital filter realizations," *Proc. IEEE* (Lett.), vol. 57, pp. 1181–1183, June 1969.

[7] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, 1970.

[8] B. Liu and T. Kaneko, "Error analysis of digital filters realized with floating-point arithmetic," *Proc. IEEE*, vol. 57, pp. 1735–1747, Oct. 1969.

[9] A. V. Oppenheim, "Realization of digital filters using block-floating-point arithmetic," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 130–136, Jan. 1970.

[10] R. B. Blackman, *Linear Data-Smoothing and Prediction in Theory and Practice*. Reading, Mass.: Addison-Wesley, 1965.

[11] L. Jackson, "An analysis of limit cycles due to multiplication rounding in recursive digital filters," in *Proc. 7th Allerton Conf. Circuit and System Theory*, pp. 69–78, 1969.

[12] S. R. Parker and S. F. Hess, "Limit-cycle oscillations in digital filters," *IEEE Trans. Circuit Theory*, vol. CT-8, pp. 687–697, Nov. 1971.

[13] P. M. Ebert, J. E. Mazo, and M. C. Taylor, "Overflow oscillations in digital filters," *Bell Syst. Tech. J.*, vol. 48, pp. 2999–3020, 1969.

[14] J. F. Kaiser, "Some practical considerations in the realization of linear digital filters," in *Proc. 3rd Allerton Conf. Circuit and Systems Theory*, pp. 621–633, 1965.

15] C. M. Rader and B. Gold, "Effects of parameter quantization on the poles of a digital filter," *Proc. IEEE* (Lett.), vol. 55, pp. 688–689, May 1967.

[16] E. Avenhaus, "An optimization procedure to minimize the word length of digital filter coefficients," presented at the London Conf. on Digital Filtering, Aug. 31, 1971.

[17] E. Avenhaus and W. Schuessler, "On the approximation problem in the design of digital filters with limited wordlength," *Arch. Elek. Übertragung*, vol. 24, pt. 12, pp. 571–572, 1970.

[18] K. Steiglitz, "Designing short-word recursive digital filters," in *Proc. 9th Allerton Conf. Circuit and System Theory* (Monticello, Ill.), pp. 778–788, Oct. 1971.

[19] A. Fettweis, "Some principles of designing digital filters imitating classical filter structures," *IEEE Trans. Circuit Theory* (Corresp.), vol. CT-18, pp. 314–316, Mar. 1971.

[20] J. Bingham, "A new type of digital filter with a reciprocal ladder configuration," in *Proc. 1970 IEEE Int. Symp. Circuit Theory, Dig. of Tech. Papers*, pp. 129–130.

[21] R. Crochiere, "Digital ladder filter structures and coefficient sensitivity," M.I.T. Res. Lab. of Electronics, Quart. Progr. Rep. 103, Oct. 15, 1971.

[22] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.

[23] B. Gold and C. M. Rader, *Digital Processing of Signals*. New York: McGraw-Hill, 1969.

[24] C. J. Weinstein, "Quantization effects in digital filters," M.I.T. Lincoln Lab. Tech. Rep. 468, ASTIA Doc. DDC AD-706862, Nov. 21, 1969.

[25] P. D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 153–157, June 1969.

[26] W. M. Gentleman and G. Sande, "Fast Fourier transforms—For fun and profit," in *Proc. Fall Joint Computer Conf., AFIPS Conf. Proc.*, pp. 563–578, 1966.

[27] C. J. Weinstein, "Roundoff noise in floating point fast Fourier transform computation," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 209–215, Sept. 1969.

[28] T. Kaneko and B. Liu, "Accumulation of roundoff error in fast Fourier transforms," *J. Ass. Comput. Mach.*, vol. 17, pp. 637–654, Oct. 1970.

[29] D. W. Tufts, H. S. Hersey, and W. E. Mosier, "Effects of FFT coefficient quantization on bin frequency response," *Proc. IEEE* (Lett.), vol. 60, pp. 146–147, Jan. 1972.

[30] L. B. Jackson, "Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 107–122, June 1970.

[31] L. B. Jackson, "An analysis of roundoff noise in digital filters," Sc.D. dissertation, Stevens Inst. Technol., Dep. Elec. Eng., Castle Point, Hoboken, N. J., 1969.

[32] A. V. Oppenheim and C. Weinstein, "A bound on the output of a circular convolution with application to digital filtering," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 120–124, June 1969.

Bibliography

[33] J. E. Bertram, "The effect of quantization in sampled-feedback systems," *AIEE Trans. (Appl. Industry)*, vol. 77, pp. 177–182, July 1958.

[34] F. Bonzanigo, "Constant-input behavior of recursive digital filters," presented at IEEE Arden House Workshop in Digital Filtering, Harriman, N. Y., Jan. 1970.

[35] R. B. Blackman, *Linear Data-Smoothing and Prediction in Theory and Practice*. Reading, Mass.: Addison-Wesley, 1965, pp. 75–79.

[36] E. E. Curry, "The analysis of round-off and truncation errors in a hybrid control system," *IEEE Trans. Automat. Contr.* (Short Papers), vol. AC-12, pp. 601–604, Oct. 1967.

[37] L. D. Divieti, C. M. Rossi, R. M. Schmid, and A. E. Vereschkin, "A note on computing quantization errors in digital control systems," *IEEE Trans. Automat. Contr.* (Corresp.), vol. AC-12, pp. 622–623, Oct. 1967.

[38] R. Edwards, J. Bradley, and J. Knowles, "Comparison of noise performances of programming methods in the realization of digital filters," in *Proc. 1969 Polytech. Inst. Brooklyn Symp. on Computer Processing in Communications*.

[39] A. Fettweis, "A general theorem for signal-flow networks, with applications," *Arch. Elek. Übertragung*, vol. 25, pp. 557–561, Dec. 1969.

[40] W. A. Gardner, "Reduction of sensitivities in sampled-data filters," *IEEE Trans. Circuit Theory* (Corresp.), vol. CT-17, pp. 660–663, Nov. 1970.

[41] B. Gold and L. Rabiner, "Analysis of digital and analog formant synthesizers," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 81–94, Mar. 1968.

[42] B. Gold and C. M. Rader, *Digital Processing of Signals*. New York: McGraw-Hill, 1969.

[43] R. M. Golden and S. A. White, "A holding technique to reduce number of bits in digital transfer functions," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 433–437, Sept. 1968.

[44] D. Herrmann and W. Schuessler, "On the accuracy problem in the design of nonrecursive digital filters," *Arch. Elek. Übertragung*, vol. 24, pt. 11, pp. 525–526, 1970.

[45] J. F. Kaiser, "Digital filters," in F. F. Kuo and J. F. Kaiser, Eds., *System Analysis by Digital Computer*. New York: Wiley, 1966, ch. 7.

[46] T. Kaneko and B. Liu, "Roundoff error of floating-point digital filters," in *Proc. 6th Allerton Conf. on Circuit and System Theory*, pp. 219–227, Oct. 1968.

[47] J. Katzenelson, "On errors introduced by combined sampling and quantization," *IRE Trans. Automat. Contr.*, vol. AC-7, pp. 58–68, Apr. 1962.

[48] W. C. Kellogg, "Information rates in sampling and quantization," *IEEE Trans. Informat. Theory*, vol. IT-13, pp. 506–511, July 1967.

[49] J. B. Knowles and R. Edwards, "Simplified analysis of computational errors in a feedback system incorporating a digital computer," presented at S.I.T. Symp. on Direct Digital Control, London, England, Apr. 22, 1965.

[50] ——, "Complex cascade programming and associated computational errors," *Electron. Lett.*, vol. 1, pp. 160–161, Aug. 1965.

[51] ——, "Finite word-length effects in a multirate direct digital control system," *Proc. Inst. Elec. Eng.*, vol. 112, pp. 2376–2384, Dec. 1965.

[52] J. B. Knowles and E. M. Olcayto, "Coefficient accuracy and digital filter response," *IEEE Trans. Circuit Theory*, vol. CT-15, pp. 31–41, Mar. 1968.

[53] A. A. Kosyakin, "The statistical theory of amplitude quantization," *Avtomat. Telemekh.*, vol. 22, p. 722, 1969.

[54] I. M. Langenthal, "Coefficient sensitivity and generalized digital filter synthesis," in *EASCON 1968 Rec.*, pp. 386–392, 1968.

[55] C. E. Maley, "The effect of parameters on the roots of an equation system," *Comput. J.*, vol. 4, pp. 62–63, 1961–1962.

[56] P. E. Mantey, "Eigenvalue sensitivity and state-variable selection," *IEEE Trans. Automat. Contr.*, vol. AC-13, pp. 263–269, June 1968.

[57] R. K. Otnes and L. P. McNamee, "Instability thresholds in digital filters due to coefficient rounding," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 456–463, Dec. 1970.

[58] C. Rader and B. Gold, "Digital filter design techniques in the frequency domain," *Proc. IEEE*, vol. 55, pp. 149–171, Feb. 1967.

[59] Q. I. Rahman, "The influence of coefficients on the zeros of polynomials," *J. London Math Soc.*, vol. 36, pt. 1, pp. 57–64, Jan. 1961.

[60] I. W. Sandberg, "Floating-point-roundoff accumulation in digital filter realization," *Bell Syst. Tech. J.*, vol. 46, pp. 1775–1791, Oct. 1967.

[61] ——, "A theorem concerning limit cycles in digital filters," in *Proc. 7th Ann. Allerton Conf. Circuit and System Theory*, pp. 63–68, 1968.

[62] J. B. Slaughter, "Quantization errors in digital control systems," *IEEE Trans. Automat. Contr.*, vol. AC-9, pp. 70–74, Jan. 1964.

[63] O. Sornmoonpin, "Investigation of quantization errors," M.Sc. thesis, Univ. of Manchester, England, 1966.

[64] T. G. Stockham, Jr., "A-D and D-A converters: Their effect on digital audio fidelity," presented at 41st Meeting of the Audio Engineering Society, New York, Oct. 5–8, 1971.

[65] D. W. Tufts, W. Knight, and D. W. Rorabacher, "Effects of quantization and sampling in digital correlators and in power spectral estimation," *Proc. IEEE* (Lett.), vol. 56, pp. 79–82, Jan. 1969.

[66] D. W. Tufts, D. W. Rorabacher, and W. E. Mosier, "Designing simple, effective digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 142–158, June 1970.

[67] C. Weinstein, "Quantization effects in frequency sampling filters," in *NEREM Rec.*, p. 222, 1968.

[68] B. Widrow, "A study of rough amplitude quantization by means of Nyquist sampling theory," *IRE Trans. Circuit Theory*, vol. CT-3, pp. 266–276, Dec. 1956.

[69] J. H. Wilkinson, "Error analysis of floating-point comparison," *Numerisch. Math.*, vol. 2, pp. 319–340, 1960.

[70] ——, *Rounding Errors in Algebraic Processes*. Englewood Cliffs, N. J.: Prentice-Hall, 1963.

[71] E. P. F. Kan and J. K. Aggarwal, "Error analysis of digital filter employing floating-point arithmetic," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 678–686, Nov. 1971.

Reprinted from the PROCEEDINGS OF THE IEEE
VOL. 60, NO. 8, AUGUST, 1972
pp. 957-976